Sustaining software-based art

Conservation strategies and institutional requirements

Röck, C.

Publication date
2024

Document Version
Final published version

Link to publication

Citation for published version (APA):
Röck, C. (2024). *Sustaining software-based art: Conservation strategies and institutional requirements*. [Thesis, fully internal, Universiteit van Amsterdam].

Sustaining Software-Based Art
Conservation Strategies and Institutional Requirements

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op dinsdag 2 juli 2024, te 13.00 uur

door Claudia Röck
geboren te Zug

Promotiecommissie

Promotor:	prof. dr. J.J. Noordegraaf	Universiteit van Amsterdam
Copromotor:	dr. K. Rechert	Hochschule Kehl - University of Applied Sciences
Overige leden:	prof. dr. M.R. van Bommel	Universiteit van Amsterdam
	prof. dr. P. Laurenson	University College London
	prof. dr. R. van de Vall	Universiteit Maastricht
	dr. S. Stigter	Universiteit van Amsterdam
	dr. A. Dekker	Universiteit van Amsterdam

Faculteit der Geesteswetenschappen

The research of this doctoral thesis received financial assistance from the Innovative Training Network New Approaches to Conservation of Contemporary Art (NACCA). NACCA is funded by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement nº 642892.

Samenvatting:

Het Behoud van op Software Gebaseerde Kunst

Behoudsstrategieën en Institutionele Vereisten

Op software gebaseerde kunst is onderhevig aan de snelle cycli van updates en vervangingen die alle software met zich meebrengt en aan de relatief korte levensduur van elektronische hardware. Het onderhouden en behouden van dergelijke kunstwerken vormt een voortdurende uitdaging voor hedendaagse kunstinstellingen. Helaas ontbreekt het bij conserveringsmaatregelen vaak aan een langetermijnperspectief. Zo vermindert het vervangen van verouderde software bijvoorbeeld niet noodzakelijkerwijs de behoefte voor toekomstige conservering, omdat elke nieuwe software zelf voorbestemd is om verouderd te raken. Dergelijke updates stellen het probleem alleen maar uit en zijn daarom niet duurzaam. Hoewel restauratoren ernaar streven om wijzigingen aan een kunstwerk tot een minimum te beperken, resulteert elke restauratiecyclus in wijzigingen. Dit proefschrift probeert een theoretisch raamwerk te ontwikkelen voor het langdurig behoud van op software gebaseerde kunst binnen instellingen, dat wordt geleid door het principe van het minimaliseren van wijzigingen aan het kunstwerk.

Om dit te bereiken stel ik een raamwerk voor dat uit drie dimensies bestaat: ten eerste stelt het criteria vast voor duurzaam behoud. Ten tweede verklaart het de impact van digitale materialiteit op de significante eigenschappen van een kunstwerk. Ten slotte conceptualiseert het het kunstwerk als een socio-technisch systeem om conserveringsstrategieën te identificeren. De duurzaamheidscriteria zijn gebaseerd op de sociale en natuurwetenschappen maar ook op het gebied van de digitale bewaring. Zij vormen drie fundamentele pijlers: conserveringsstrategieën, *network of care* (een term bedacht door Annet Dekker (2014)) en instelling. Het begrip digitale materialiteit komt voort uit het onderzoek van Paul Dourish (2017), die het verband benadrukt tussen softwaretechnologie en onze perceptie en gebruik van informatie. Dit concept van digitale materialiteit leek mij belangrijk omdat het gemakkelijk is om de relevantie van de software voor een kunstwerk over het hoofd te zien aangezien het feit dat software vaak niet op het eerste gezicht zichtbaar is.

Andrew Feenbergs (2017) concept van het technosysteem biedt een theoretisch kader voor de analyse van op software gebaseerde kunst, dat nuttig blijkt bij het identificeren van duurzame conserveringsstrategieën. Het kunstwerk als socio-technisch

systeem wordt beïnvloed door markten, technologieën en administraties. Dit zogenaamde *technosysteem* bestaat uit verschillende lagen, bouwstenen en de omgeving van het kunstwerk. De technische lagen zijn vaak het uitgangspunt voor conserveringsstrategieën. De bouwstenen zijn zo gedefinieerd dat voor elke bouwsteen een aparte conserveringsstrategie kan worden toegepast. Het identificeren van de omgeving van het kunstwerk is belangrijk omdat het vaak bijdraagt aan defecten en een eigen conserveringsstrategie kan vereisen.

Dit proefschrift houdt zich bezig met conserveringsstrategieën op softwareniveau en gaat niet in op het behoud van hardware. Daarnaast hanteert het een vrij enge definitie van conservering, waardoor strategieën zoals herinterpretatie of het voortzetten van een procesmatig kunstwerk in dit proefschrift worden uitgesloten van het duurzaamheidsonderzoek. Tot slot onderzoek ik geen duurzaamheidskwesties met betrekking tot de bescherming van de natuurlijke omgeving of klimaatverandering.

Om het theoretisch kader te testen, heb ik twee casestudies uitgevoerd. Voor elke case heb ik twee conserveringsstrategieën toegepast - migratie en emulatie - en deze vergeleken op duurzaamheid en authenticiteit van het kunstwerk. De eerste casestudy, *Horizons* (2008) van Geert Mul, is een generatieve video-installatie die reageert op een sensorinput en biedt een kans om de voor- en nadelen van emulatie versus migratie te bespreken. Het behoort toe aan een middelgroot kunstmuseum, het Museum Boijmans Van Beuningen in Rotterdam. *TraceNoizer* (2001-2004), van de kunstenaarsgroep LAN, is een webgebaseerd kunstwerk dat illustreert hoe de internetomgeving van een kunstwerk kan worden gestabiliseerd en geïntegreerd als onderdeel van de conservering ervan. Het is eigendom van het House of Electronic Arts, een kleine instelling gespecialiseerd in digitale kunst.

De casestudies tonen de waarde van het theoretische kader aan. Bepaalde praktische aspecten, zoals de slechte geluidskwaliteit van de geëmuleerde versie van *Horizons*, vereisen echter compromissen die de duurzaamheid en/of de belangrijke eigenschappen van een kunstwerk beïnvloeden. Het toepassen van het theoretisch kader dat hier naar voren is gebracht, moedigt een systematische benadering van conservering aan en resulteert zo in conserveringsstrategieën die voor het grootste deel duurzamer zijn voor zowel het kunstwerk als de instelling. Daarnaast identificeert dit proefschrift de voorwaarden die instellingen nodig hebben om op software gebaseerde kunst te behouden. Bovendien verduidelijkt het bepaalde definities van conserveringsstrategieën door digitale emulatie te categoriseren als een inkapselingsstrategie, en reconstructie- en documentatiestrategieën toe

te voegen aan het bestaande repertoire van conserveringsstrategieën. Het bijgewerkte repertoire bestaat dus uit opslag, migratie, inkapseling, reconstructie en documentatie. Deze verduidelijkingen zijn noodzakelijk: terwijl softwaregebaseerde kunst op het snijvlak ligt van digitale preservering en het behoud van hedendaagse kunst, bestaan er discrepanties in de respectievelijke definities van conserveringsstrategieën binnen deze twee gebieden. Een gemeenschappelijk vocabulaire maakt het veel gemakkelijker om conserveringsstrategieën en ethische overwegingen tussen deze gebieden te vergelijken.

Dit proefschrift benadrukt ook het potentieel van substrategieën om onvermijdelijke compromissen in conservering te compenseren. Het combineren van conserveringsstrategieën is een andere effectieve aanpak voor het bereiken van duurzame conserveringsoplossingen. Ten slotte kan het voor kleinere instellingen met beperkte conserveringsbudgetten de moeite waard zijn om de belangrijke eigenschappen van een kunstwerk aan te passen om externe afhankelijkheden te verminderen. Het verlies van functionaliteit van het kunstwerk - wat ik "achterwaartse veranderingen" noem - kan een acceptabele afweging zijn in ruil voor een realistische langetermijnbenadering van conservering.

Summary:
Sustaining Software-Based Art
Conservation Strategies and Institutional Requirements

Software-based art is subject to the rapid cycles of updates and replacements common to all software, as well as the relatively short life expectancy of electronic hardware. Maintaining and preserving such artworks poses a constant challenge for contemporary art institutions. Unfortunately, preservation measures often lack long-term perspectives. For example, replacing outdated software does not necessarily reduce the need for future conservation, because any new software is itself destined to become obsolete. Such updates simply defer the problem and are therefore not sustainable. Although conservators strive to minimise alterations made to an artwork, every conservation cycle results in modifications. This dissertation seeks to develop a theoretical framework for the long-term preservation of software-based art within institutions that is guided by the principle of minimising changes to the artwork.

To achieve this, I propose a framework comprising three dimensions: firstly, it establishes criteria for sustainable preservation. Secondly, it explains the impact of digital

materiality on the significant properties of an artwork. Finally, it conceptualises the artwork as a socio-technical system in order to identify conservation strategies. Sustainability criteria are informed by the social and natural sciences, as well as digital preservation, and constitute three fundamental pillars: conservation strategies, network of care (a term coined by Annet Dekker (2014)), and institution. The notion of digital materiality stems from the research of Paul Dourish (2017), who emphasises the connection between software technology and our perception and utilisation of information. This concept of digital materiality seemed important to me as it is easy to overlook the relevance of the software for an artwork due to the fact that the software is often not visible at first sight.

Andrew Feenberg's (2017) concept of the technosystem presents a theoretical framework for the analysis of software-based art which proves useful in identifying sustainable conservation strategies. The artwork as a socio-technical system is influenced by markets, technologies, and administrations. This so called *technosystem* consists of various layers, building blocks, and the artwork's environment. Its technical layers are often the starting point for conservation strategies. The building blocks are defined in such a way that a distinct conservation strategy can be employed for each. For its part, identifying the artwork environment is important as it frequently contributes to malfunctions and may require its own conservation strategy.

This dissertation is concerned with conservation strategies at the software level and does not address hardware preservation. Additionally, it applies a fairly narrow definition of conservation, excluding strategies such as reinterpretation or the continuation of a processual artwork from the sustainability research in this dissertation. Lastly, I do not explore sustainability issues relating to the protection of the natural environment or climate change.

To test the theoretical framework, I conducted two case studies. For each, I applied two preservation strategies – migration and emulation – and compared them in terms of their sustainability and the authenticity of the artwork. The first case study, *Horizons* (2008) by Geert Mul, is a generative video installation that reacts to a sensor input, and presents an opportunity to discuss the merits and challenges of emulation versus migration. It belongs to a medium-sized art museum, the Museum Boijmans Van Beuningen in Rotterdam. *TraceNoizer* (2001-2004), by the artist group LAN, is a web-based artwork that illustrates how an artwork's internet environment can be stabilised and integrated as part of its preservation. It is owned by the House of Electronic Arts, a small institution specialising in digital art.

The case studies demonstrate the value of the theoretical framework. However, certain practicalities such as the poor sound quality of the emulated version of *Horizons* require compromises that impact sustainability and/or the significant properties of an artwork. Implementing the theoretical framework put forward here encourages a systematic approach to conservation, and thus results in conservation strategies that are, for the most part, more sustainable for both the artwork and the institution. In addition, this dissertation identifies the conditions necessary for institutions to sustain software-based artworks. As another outcome, it clarifies certain definitions of conservation strategies by categorising digital emulation as an encapsulation strategy, and adding reconstruction and documentation strategies to the existing repertoire of conservation strategies. The updated repertoire thus consists of storage, migration, encapsulation, reconstruction, and documentation. These clarifications are necessary: whilst software-based art lies at the intersection of digital preservation and contemporary art conservation, there exist discrepancies in the respective definitions of conservation strategies within these two fields. A common vocabulary makes it far easier to compare conservation strategies and ethical considerations across these fields.

This dissertation also highlights the potential of sub-strategies to counterbalance compromises in conservation when these are unavoidable. Combining conservation strategies presents another effective approach to achieving sustainable conservation solutions. Finally, for smaller institutions with restricted conservation budgets, modifying an artwork's significant properties to reduce its external dependencies can prove worthwhile. The loss of artwork functionality – which I call backward changes – can be an acceptable trade-off in exchange for a realistic long-term approach to preservation.

Contents

Acknowledgements

This dissertation was made possible by a set of lucky circumstances, such as the fact that I could allow myself to live off a modest income for a few years, and that I had the support of those around me.

During my research and writing process, I was helped and accompanied by my PhD supervisors, Professor Julia Noordegraaf, who specialises in Digital Heritage at the University of Amsterdam, and Professor Klaus Rechert, who is an expert in Information Technology at Kehl University (DE). They offered valuable feedback and guidance throughout the process. Julia Noordegraaf gave me important advice on structuring a dissertation and building a chain of arguments in the humanities. Klaus Rechert introduced me to the variants of emulation, and gave me critical feedback from a computer science perspective. Along with his collaborator Rafael Gieschke, he instituted a preservation strategy for *TraceNoizer*, building a virtual network of the artwork's components and its environment.

Gaby Wijers, director of LI-MA, a platform dedicated to research into, distribution, and the preservation of media art, facilitated the case study of *Horizons* and provided a professional conservation workspace in which to carry out conservation measures. Geert Mul, the artist behind *Horizons* and *Shan Shui* (2013), shared crucial information about these artworks in a number of interviews and encounters during the process of preparing Mul's solo exhibition *Match Maker* in 2016. He paid me to document and advise on the conservation of *Shan Shui* in the run-up to its permanent display at Dortmunder U in 2020. Carlo Prelz, the programmer involved in *Horizons* and *Shan Shui*, explained to me how he had programmed the software for these artworks. I also worked with the programmer Teal Gaure, who helped me to set up the emulation for *Horizons*, recompiled QEMU to enable paravirtualisation, and who programmed a number of preservation features such as making the sensor input human-readable and facilitating screencasts. He also set up the packaging system Nix for *Shan Shui* at the Dortmunder U in 2020 which allows the conservator to initiate a new instance of *Shan Shui* with the same software parameters. Sandra Kisters, the head of collections, and Roelie Zijlstra, the registrar, at the Museum Boijmans Van Beuningen, kindly allowed me to interview them about the acquisition of *Horizons*.

Sabine Himmelsbach, director of the House of Electronic Arts in Basel, provided me access to the case study *TraceNoizer* and allowed me to use all of the information about the artwork in the conservation file. Annina Rüst, Roman Abt, Fabian Thommen, Marc Lee, and

Silvan Zurbruegg made up the temporary group LAN who created *TraceNoizer*. Three of these artists, Annina Rüst, Fabian Thommen, and Marc Lee, accorded me an interview about the meaning and conservation of the work. Fabian Thommen carried out the migration strategy for *TraceNoizer,* which was paid for by the House of Electronic Arts.

My partner, Gregor Bossert, gave me invaluable emotional support and accepted my decision to live in Amsterdam for the last few years. He always showed interest in my dissertation and provided constructive feedback for the duration of the process.

My research was part of and funded by the NACCA (New Approaches in the Conservation of Contemporary Art) project, one of the European Union's Marie Skłodowska-Curie Innovative Training Networks. In the network's summer and winter schools we received training to enhance our academic skills. My peers, other doctoral researchers from the NACCA project such as Aga Wielocha and Dušan Barok, provided real moral support and remain part of my network. Along with Brian Castriota, they influenced my thoughts about conservation, documentation, and the question of authenticity in contemporary art.

I would like to gratefully acknowledge Karin Christof, a fellow PhD candidate and my neighbour at the University of Amsterdam, for enduring my fluctuating moods and encouraging me to complete my project. I am also grateful to Fan Yang, another PhD candidate who brought light to my repetitive days during the pandemic.

Simon Ferdinand proofread the dissertation to make it more enjoyable to read.

Finally, I tried to bypass the overwhelming monopoly of Adobe and its vicious licensing subscriptions, and made the graphics in this dissertation with the free software yEd and Libre Office Draw (open-source).

I am grateful to everyone who has supported me on my PhD journey!

Co-Authored Articles

Roeck, Claudia, Klaus Rechert, and Julia Noordegraaf. 2018. "Evaluation of Preservation Strategies for an Interactive, Software-Based Artwork with Complex Behavior Using the Case Study *Horizons* (2008) By Geert Mul." In *IPres 2018*, edited by iPres 2018. https://osf.io/y3gcu and https://dare.uva.nl/search?identifier=4880bd3e-d3ba-4b16-9ead-a4ff77db2f0a

Chapter 4 (section 4.1 to 4.8 and section 4.10) was published as a peer-reviewed conference paper at the digital preservation conference iPres in 2018. The paper describes the work that I did on my first case study, *Horizons* by Geert Mul, for which Klaus Rechert shared important insights regarding the potential of paravirtualisation. The writing is my own but I received important corrections and improvements from Julia Noordegraaf and Klaus Rechert through several revisions. The paper was then peer reviewed with little feedback. It was my first piece of writing about the sustainability of conservation strategies. In the meantime, I further developed the theoretical framework (see Chapter 3) and definitions (see section 6.3 and the glossary).

Roeck, Claudia, and Teal Gaure. 2021a. "Wiki for the Documentation of the Permanent Installation of Shan Shui at the Dortmunder U." Accessed November 24, 2023. https://gitlab.com/shanshui/boij/-/wikis/home.

Section 4.9 is based on the Wiki Documentation of the Permanent Installation of Shan Shui at the Dortmunder U. I wrote section 4.9 three years after the above-mentioned iPres-article. When Geert Mul installed *Shan Shui*, an artwork based on the same software as *Horizons* at the Dortmunder U in 2020, I advised the artist regarding conservation strategies and documented the artwork online. The programmer Teal Gaure recommended Nix as a package manager and carried out the conservation measures. Section 4.9 is based on the Wiki documentation that I wrote for *Shan Shui* (Roeck and Gaure 2021a). I added Teal Gaure as a co-author because of his active role in the conservation of Shan Shui at Dortmunder U. The Wiki is not peer reviewed, and is intended to serve the Dortmunder U and the artist as a (technical and conservation) documentation of the artwork.

Roeck, Claudia, Klaus Rechert, Julia Noordegraaf, and Rafael Gieschke. 2019. "PRESERVATION STRATEGIES for an INTERNET-BASED ARTWORK YESTERDAY, TODAY and TOMORROW." In *IPres2019: Conference Proceedings*, edited by iPres2019, 179–90. Amsterdam. https://ipres2019.org/static/pdf/iPres2019_paper_125.pdf and https://hdl.handle.net/11245.1/3b90f2b6-d6c0-4fc0-adc9-63c166f23307.

Chapter 5 (section 5.1 to 5.8 and 5.10) was published as a peer-reviewed conference paper at the digital preservation conference iPres in 2019, and is based on a paper written with Rafael Gieschke about my second case study, *TraceNoizer* by LAN. Before we wrote this paper, I transferred the Linux Live CD of *TraceNoizer* to Klaus Rechert and Rafael Gieschke, for them to experiment with Emulation-as-a-Service. They managed to link several emulators to build an emulator network for *TraceNoizer*, consisting of several networked components such as the Internet Archive replacing the Internet, and the emulated back and front-end of *TraceNoizer*. Rechert and Gieschke therefore wrote Chapter VII of the paper (section 5.7 in this dissertation) about their results, whilst I wrote the rest (sections 5.1 to 5.6, 5.8 and 5.10) and designed the graphics. Julia Noordegraaf and Klaus Rechert revised my chapter several times and offered feedback with corrections and improvements. The paper was peer reviewed with little feedback.

The article focused on conservation strategies but did not consider *TraceNoizer*'s display history or the potential of adapting its display to the exhibition context without impeding its long-term preservation. To complete the case study, I added section 5.9 on *TraceNoizer*'s display history to this dissertation. In order for the paper to fit in my dissertation, I had to adapt certain terms: I replaced the term simulation with reconstruction and made some other small adaptations. Otherwise, the article is reproduced verbatim.

Roeck, Claudia. 2023. "Languages of Conservation: A Comparison Between Internet-Based Art and Built Heritage." In Conservation of Contemporary Art: Bridging the Gap Between Theory and Practice, edited by Renée Van de Vall and Vivian van Saaze. Studies in Art, Heritage; Law and the Market 9: Springer. https://link.springer.com/book/10.1007/978-3-031-42357-4

This paper is part of a collection of articles written within the framework of the NACCA project. I wrote it alone and adapted it according to two anonymous peer reviews. Renée van de Vall offered useful advice on implementing reviewer feedback.

I did not quote this article verbatim, but integrated its ideas in my dissertation. Moreover, I not only used the example of *TV Bot* (2004) by Marc Lee in the paper, but in this dissertation too (primarily Chapter 1, and in references in Chapters 6 and 7). The same goes for the internet-based *TraceNoizer*, which was not only a case study in the iPres paper, but also in the article "Languages of Conservation: A Comparison Between Internet-Based Art and Built Heritage" (Roeck 2023).

1 Introduction[1]

"A newer version is available. Would you like to update?" We have all seen such a message, nudging us to update software when we open our computer. In fact, it is increasingly common for the software provider to no longer ask, and to simply update automatically. Then there are those cases of software that is not even installed on one's own computer, but on a remote server to be accessed through a web browser or local client software with an internet connection. Usually, software updates bring new features, and discontinue older ones. What, then, to do if a project depends on a discontinued feature? What if the project can still run with the updated software, but it looks or behaves differently? These questions come up in our everyday exposure to technology. However, imagine the effects on a project with a purpose other than a very specific set of tasks, such as accounting software where the look and feel is relatively unimportant. Imagine, for instance, that the project affected is an artwork with a very particular look and feel. Or an artwork based on a specific input with interactive components. What does it mean for the artwork if the software is updated, perhaps multiple times? What will it look like, and will it still function in the same way? How many times can the software be updated before the artwork is fundamentally changed? What follows illustrates the potential dilemmas involved in preserving software-based art.

TV Bot 1.0 by Marc Lee, an internet-based artwork that ran through a web browser, underwent a metamorphosis due to its preservation. *TV Bot 1.0* presents "world news as soon as it happens" (Lee 2004c) by collecting news from different internet news agencies. Lee created the first version in 2004, when the idea of news coverage that was more up-to-date than on television was still unthinkable (see Figure 1 and Figure 2). The website that hosted *TV Bot 1.0*, however, was internally connected to numerous internet news agencies, webcams, and internet radios; moreover, it was based on a video and audio format (RealMedia) that these news sources used to distribute the news via the internet. After a few years, some of the news sources began to replace RealMedia with the Flash format, whilst

[1] This chapter is based on
Roeck, Claudia. 2023. "Languages of Conservation: A Comparison Between Internet-Based Art and Built Heritage." In Conservation of Contemporary Art: Bridging the Gap Between Theory and Practice, edited by Renée Van de Vall and Vivian van Saaze. Studies in Art, Heritage; Law and the Market 9: Springer. https://link.springer.com/book/10.1007/978-3-031-42357-4

others ceased operations or changed URL. Hence, *TV Bot 1.0* increasingly showed error pages instead of the news. Consequently, Lee had to update *TV Bot 1.0* with the Flash format and adapt the list of news sources, which resulted in *TV Bot 2.0* (see Figure 3 and Figure 4). This was part of the 2010 research project "Digital Art Conservation" (École supérieure des arts décoratifs (esads), Strasbourg et al. 2010 and Serexhe 2013), which focused on the question of conserving computer-based art. With the restoration, Lee gave the artwork a makeover: instead of an introductory text that met the user upon their accessing the website, he inserted a map with an overview of the locations of the TV stations that *TV Bot* drew on. He also introduced a Twitter feed on a blue background, added the language of the radio or TV station, and inserted a *TV Bot 2.0* logo in the bottom right-hand corner.

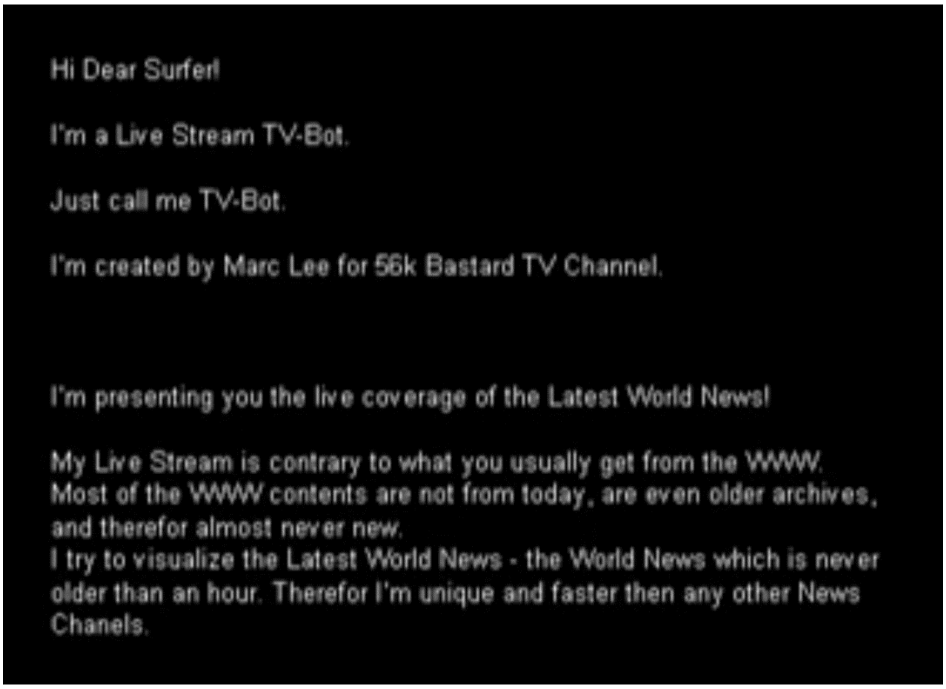


Figure 1: *TV Bot 1.0* (2004) by Marc Lee (screenshot from screencast, Lee 2004b). Introductory text on website landing page.

Figure 2: *TV Bot 1.0* (2004) by Marc Lee (screenshot from screencast, Lee 2004a)

Figure 3: *TV Bot 2.0* (2010) by Marc Lee (screenshot from screencast, Lee 2010). Map with locations of news sources on website landing page.

Figure 4: *TV Bot 2.0* (2010) by Marc Lee (screenshot from screencast, Lee 2010). Twitter feed (blue background), language metadata, and *TV Bot* 2.0 logo added.

Due to the obsolescence of the Flash format, Lee had to update the work again in 2016 (see Figure 5). He replaced all of the internet news agencies and radio stations with Twitter feeds and introduced an interactive component that allowed the user to enter one or several search terms to select tweets. The video content is streamed from YouTube. The language metadata has disappeared, and the look and feel are quite different, due to the pink background colour of the metadata and logo in contrast to the green and red backgrounds of versions 1.0 and 2.0. The content of *TV Bot 3.0* is less made up of general news, but is now more personal, due to the fact that the specific news channels have been replaced with social media.

The metamorphosis of *TV Bot* is most likely not over yet. The piece is an example of an artwork that is maintained and developed by the artist themself. Lee also made screencasts of the discontinued versions of *TV Bot*, which are the only existing documentary evidence of their existence. This raises questions: would an institution that buys such an artwork be allowed to make changes as far-reaching as Lee did, in the potential absence (whether through death or indifference) of the artist? Would an institution be financially able to repeat such cycles of conservation every five to six years? Would it be able to maintain and monitor such an artwork in between conservation cycles? Would it be able to

provide and maintain the necessary infrastructure, such as a web server in this case? What are sustainable conservation strategies from an institutional point of view?

Figure 5: *TV Bot 3.0* (2016) by Marc Lee (screenshot taken from the website in 2021 before Twitter became X, Lee 2021a).

Before the outbreak of Covid-19, contemporary art museums only sporadically acquired software-based artworks. Whilst the pandemic accelerated the digitisation process of cultural institutions to increase accessibility during lockdowns (Burke, D. Jørgensen, and F. A. Jørgensen 2020, 117), it remains to be seen whether it also accelerated the acquisition of software-based artworks. Indeed, it had previously been relatively unusual for a museum to acquire and preserve such artworks. Whilst time-based media artworks such as video and film may represent a small percentage of a collection, most contemporary art museums own no more than ten software-based artworks, which make up less than one per cent of their collection. Due to the fact that many galleries do not employ conservators of time-based media, they are often without expertise when it comes to the conservation of software-based art, and their capacity to accrue knowledge in this area is also therefore limited. Moreover, there was no systematic approach or guidelines as to how to preserve software-based artworks in the long term until the publication of the book *Conservation of Time-Based Media Art* (Phillips and Engel 2023). The conservation strategies compiled by the Variable Media Network in 2004 were designed for variable contemporary art in general and are not sufficiently geared towards digital preservation. The advice given on Matters in Media Art's website (MoMA et al. n.d.) focuses on storage strategies and the requirements of a digital

archive, but does not provide advice on the implementation of sustainable conservation strategies. Furthermore, in practice, short-term considerations such as the preparation of an artwork for an exhibition are typically those that lead conservation measures. Strategic long-term approaches for the conservation of software-based art, meanwhile, are rare.

The rapid obsolescence of software-based art and its short conservation cycles force us to think beyond single conservation treatments. It is therefore the goal of this dissertation to interrogate how software-based artworks can be sustained in the long term in an institutional context. My investigation bears on what sustainable means for the conservation of software-based art in institutions, what conservation strategies are available, and how they can contribute to sustaining an artwork in the long term. I look specifically at those conservation strategies that are available for the medium software, and at the need to analyse software-based artworks in order to identify appropriate strategies.

Theoretical Framework: The Artwork as Technosystem

To answer these questions, I propose a theoretical framework made up of three pillars: sustainability criteria, digital materiality (as described by Dourish 2017]), and the technosystem (as described by Feenberg 2017]). For the sustainability criteria, I draw on the fields of nature conservation and digital preservation. The question of sustainability has been much discussed in relation to protecting the natural environment since the Club of Rome published *The Limits to Growth* in 1972 (Donella Meadows et al. 1972). As sustainability entails primarily long-term perspectives, it must effect a balance between different interests. For the conservation of art, these interests include the preservation of artworks in their authentic state, sound institutional finances, the needs of other artworks in the collection, and the ability of future conservators to adapt to the emergence of new knowledge about artworks and their conservation requirements.

The second pillar – Dourish's digital materiality – is important, as an artwork's materiality is a fundamental criterion in the selection of conservation strategies. For instance, Barbara Appelbaum, a conservation scholar who has written extensively on conservation ethics, divides the information necessary to make decisions about conservation measures into material and non-material considerations (Appelbaum 2007, 12). Similarly, the revised Decision-Making Model for Contemporary Art Conservation and Presentation points to "an artwork's material aspects as well as the artwork's intangible properties" as central to such decisions (Cologne Institute of Conservation Sciences / TH Köln 2020, 2). As digital artworks are often seen to be ephemeral and hence immaterial, the risk of

underestimating their digital materiality is very real. Therefore, I consider it important to develop a concept for the digital materiality of software-based artworks.

The third pillar, the technosystem, provides a concept for the artwork and its environment. Lee's *TV Bot* illustrates perfectly how dependent an artwork is on its sociotechnical environment; in this case, external web services. The causes of damage to software-based artworks are therefore often not, or not only, material breakdown, but can be traced to a change in the artwork's sociotechnical environment. Besides, the technosystem designates a wider context: the institutional environment of software-based artworks in a museum and the corresponding network of care, a term coined by Annet Dekker (Dekker 2014, 81). The technosystem highlights the interplay between society and technology, and includes markets and administrations as fields of action (Feenberg 2017, x). According to Feenberg, "Administrations resemble science in classifying objects and treating them uniformly under rules" (Feenberg 2017, 22). In the context of this dissertation, administrations can be understood as those institutions, such as museums, that collect software-based art. Feenberg describes himself as a critical constructivist, acknowledging society's formative role in shaping the world and its technology, but also recognising the influence that cultural, economic, and political contexts exert on this process. Art conservation is, therefore, situated in this field of tension between technology, the art market, the technology market, institutions, and society. The second part of the term technosystem, that is, the "system", is important, as it invites the conservator to conceptualise a software-based artwork as a system through which preservation solutions must be identified. Such artworks are often highly complex in terms of both their software and hardware; identifying them as systems therefore affords the conservator a useful overview by which smaller details can be overlooked in favour of more important components and connections.

Methodology and Case Studies

My research into long-term perspectives for conservation strategies required me to conduct case studies, a method often used in conservation research (see, for instance Hummelen and Sillé 1999] or Scholte and Wharton 2011]). For this, I selected two artworks, which allowed me to conduct an in-depth technical analysis of each and to carry out hands-on conservation myself. I employed two different conservation strategies on each artwork. I then compared the conservation results in light of the significant properties of the artworks and based on the sustainability criteria mentioned in the theoretical framework. I

conducted interviews with artists and programmers about the meaning of the artworks, the role of the software and hardware, and how they see the artworks in the future. I also interviewed collection managers from the Museum Boijmans Van Beuningen about the acquisition and conservation of one of the artworks. In order to carry out the conservation strategies on the first case study, I requested the support of Teal Gaure, a software engineer. For the second case study, I was able to draw on the support of Rafael Gieschke and Klaus Rechert, computer scientists at the University of Freiburg. Once I had completed the case studies, I examined the application of the theoretical framework to them more thoroughly, so as to test its suitability.

Both artworks are over ten years old, which already gives a sense of the challenges for their conservation. *Horizons* (2008) by the Dutch artist Geert Mul represents the first case study. It is owned by the Museum Boijmans Van Beuningen (Rotterdam, NL) and shows the museum's painting collection in an interactive way. It consists of a computer that generates an interactive video projected onto three adjacent screens, with speakers containing a sensor that tracks visitor movements in front of the projection. Video and sound respond to the movements of visitors. Mul's *Shan Shui* (2013) uses the same software and hardware as *Horizons* (though it only has two projectors), and other image sources. As Geert Mul exhibited *Shan Shui* in 2016, and as *Shan Shui* was a case study in LI-MA's project *Future Proof* (LI-MA 2016-2017b), I included *Shan Shui* in my research of *Horizons*.

The second case study is the internet-based artwork *TraceNoizer* (2001-2004) by LAN. At the time of its launch in 2001, *TraceNoizer* was accessible via a web browser and allowed users to obfuscate their own traces on the internet by producing fake websites, so-called clones. Where *Horizons* and *Shan Shui* are software-based artworks that depend on a hardware environment, *TraceNoizer* is a key example of a software-based artwork that depends on a non-physical internet environment. This environment, consisting of other websites, web technologies, and web services, is in a constant state of change and therefore poses a risk to the long-term functioning of *TraceNoizer*, as in the case of Marc Lee's *TV Bot* discussed above. Furthermore, *TraceNoizer* is a so-called server-side website, relying not only on a user computer with a web browser for its interpretation but on computations on the web server, making its preservation even more complex.

As an interactive video installation and a web-based artwork, my case studies represent two particularly prevalent types of software-based art. However, I have designed

my theoretical framework in such a way as to guarantee the analysis of similar types of such artworks.

Delimitations

In this dissertation, I use the terms conservation and preservation interchangeably, as both fields (art conservation in museums and digital preservation in archives and libraries) intersect in software-based art conservation. Although archives and libraries usually hold digital objects that are more standardised than the software-based artworks that one may find in a museum, and although they focus more on the preservation of information rather than behaviour, both – that is, archives and libraries on the one hand and museums on the other – aim to preserve their digital objects digitally. However, I do differentiate between safeguarding and conserving as defined by Judith Butler (quoted in (Castriota 2019a, 44). For Butler, conservation/preservation "seeks to secure the life that already is" as opposed to forms of safeguarding that create the conditions for an artwork to develop further in different directions. My research, therefore, does not touch on strategies such as reinterpretation or the continuation of a processual artwork that fall under the term "safeguarding". For safeguarding, sustainability criteria other than those that I defined in the theoretical framework apply, such as for example sustainable software design (Robillard 2016).

This dissertation does not investigate the impact of software-based art conservation on the natural environment. Due to the small number of software-based artworks in museum collections, their environmental impact did not seem of major significance at this point in time.

Structure of the Thesis

Chapter 2 establishes an overview of existing research in the field of software-based art conservation. For this, I discuss the art conservation literature as well as literature in the field of digital preservation in archives and libraries. I situate software-based art in a contemporary art conservation context, examining it in terms of Dekker's research into net art conservation (Dekker 2014) and Ensom's work on documenting software-based art (Ensom 2018). Whilst Dekker highlights the networked and ambiguous character of net art, Ensom has added to this the multiplicity, complexity, and performative character of software-based art. The dependency of software-based artworks on their context and

technical environment create challenges for conservation and complicate their sustainable preservation.

Earlier theoretical approaches to the conservation of contemporary art tended to focus on the challenges posed by the variability of contemporary art installations. These included the project “Inside Installations” (2004-2007) (Scholte and Wharton 2011) and Pip Laurenson’s paper on “Authenticity, Change and Loss in the Conservation of Time-Based Media Installations” (Laurenson 2006). The variability of artworks, along with the subjective question of what we consider significant in an artwork, poses the problem of how software-based art can be preserved in the long term without having to start from scratch and redefine its significant properties after each conservation cycle.

Through the definition of conservation strategies in 2004 and the case studies linked to them, the Variable Media Network (Berkeley Art Museum / Pacific Film Archives Berkeley et al. 2001) helped to establish a set of conservation practices and ethics concerning art installations and time-based media artworks. This was a crucial contribution to contemporary art conservation, as it allowed a discussion of conserving installation art in a more strategic way, and not simply in relation to an artwork’s materiality. These definitions of conservation strategies were the point of departure for my research. They are not specifically geared towards software-based artworks, and some of the terms they use, such as emulation, have a different meaning in the field of digital preservation. This is where I identified the gap that I am exploring here. More importantly, it became clear that conservation strategies were usually only applied once to a case study. Given the short conservation cycles of software-based artworks, I am interested in the long-term effects of repeatedly applying these strategies, and how they can be applied and evaluated for software-based artworks.

Conservation theory has focused primarily on authenticity as a decision-making criterion for conservation measures. With the exception of the Cologne Institute of Conservation Sciences’ recently revised decision-making model (Cologne Institute of Conservation Sciences / TH Köln 2019, revised 2021, 14), the conservation ethics literature only rarely discusses the financial, human, and infrastructural resources necessary for conservation, although they are crucial criteria for selecting conservation measures in institutional contexts. Most importantly, sustainability criteria for the long-term preservation of artworks are missing. Therefore, in my evaluation of the long-term effects of conservation strategies for software-based artworks, I take these institutional factors into account.

Chapter 3 develops the theoretical framework discussed above. For this, I take a sidestep into science and technology studies, widely used in the field of contemporary art conservation scholarship (see, for example, Saaze [2013]), and which proves fruitful for my own research questions. Science and technology studies are concerned with the reciprocal influence of society and technology, and are therefore interdisciplinary, connecting technology studies with the humanities. This approach is well suited to my research, as conservation is a highly interdisciplinary field that sits between technology and art. The term sustainability is itself an interdisciplinary term that is used in several disciplines such as social sciences and economics.

Chapter 4 presents my analysis of the first case study, *Horizons* by Geert Mul. I conduct two conservation strategies, migration and emulation, and compare them in terms of how they affect the significant properties of the artwork and the extent to which they can be considered sustainable. Chapter 4 is based on an article published at the 2018 iPres conference. It looks at conservation strategies capable of handling graphically intensive software. Section 4.9 describes the sub-case study *Shan Shui*. I assisted Geert Mul with the permanent installation of *Shan Shui* at Dortmunder U, and we applied a conservation strategy that consisted of migration, executed within a software package management system that facilitates the reproducibility of the migration. I also documented the artwork in the same version control system where the source code is managed and distributed, which makes it easy to share as well as to adapt documentation and software.

Chapter 5 describes the second case study, *TraceNoizer* by LAN. As with the previous case study, I compare different conservation strategies to identify which is best for the artwork's long-term preservation. This chapter is based on an article published at the 2019 iPres conference. It discusses conservation strategies that are suitable for stabilising an artwork's environment, such as websites, or a web service such as Google.

Chapter 6 applies my theoretical framework to both case studies, thereby demonstrating its value for conservation practice. However, as was to be expected, compromises often have to be made in practice, either in terms of significant properties or sustainability. The chapter also argues that if conservators conceptualise artworks as consisting of building blocks, layers, and an environment, this facilitates the identification of sustainable conservation strategies. Furthermore, I contend that my theoretical framework is capable of describing not only the artworks but also the conservation strategies in a more systematic way. Moreover, chapter 6 fans out each conservation strategy into several digital sub-conservation strategies, each of which is useful in specific circumstances. The chapter

addresses solutions that bridge between long-term preservation and short-term curation, and finally discusses institutional prerequisites for the long-term conservation of software-based artworks.

Chapter 7 synthesises recommendations for the sustainable preservation of software-based art from the previous chapter, and provides an outlook for future research. Finally, it opens up the discussion to include a broader view of socio-technical developments and sustainability.

A glossary at the end of the dissertation offers definitions of terms specific to contemporary art conservation, computer science, and digital preservation.

2 Conservation of Software-Based Art: State of the Art and Definitions

This chapter prepares the ground for the theoretical framework to be developed in Chapter 3. Here I will discuss the literature relevant to the conservation of software-based art, define important terms, and identify gaps in knowledge that this thesis will strive to fill[2].

Section 2.1 positions software-based art in the field of contemporary art and offers a discussion of the characteristics of such art in relation to its conservation. As software is a crucial component of software-based art, I reflect on its properties in section 2.2, and draw on various concepts from computer science to do this. In section 2.3, I address the question of authenticity and change in relation to contemporary art in conservation theory. A number of dissertations have been written about this recently, one of them within the context of the NACCA project[3] of which this dissertation is also part. I also reflect on the concept of significant properties, often evoked in the digital preservation of archives and libraries. As software-based art sits at the intersection between the digital preservation of archives and art conservation, I discuss conservation strategies in each field separately in section 2.4. This division makes sense given that the objects of digital preservation are generally more uniform than art objects, and it is therefore easier to automate or scale preservation measures in such cases. Section 2.5 explores the criteria for conservation decisions taken in institutional contexts. I then identify the stakeholders in software-based art conservation in section 2.6. Section 2.7 builds on the previous sections to identify any knowledge gaps regarding the conservation of software-based art. Finally, I offer a brief description of my research methodology in section 2.8.

2.1 Software-Based Art and Contemporary Art

This section explores definitions of the term software-based art, and outlines the characteristics of software-based art that are relevant for its conservation. It establishes software-based art as a type of contemporary art with the ultimate goal of applying the same conservation ethics as in other forms of contemporary art conservation.

---

[2] To clarify the terms defined in this and the following chapter, I attach a glossary at the end of this dissertation.

[3] New Approaches for the Conservation of Contemporary Art (NACCA), see acknowledgments.

A straightforward definition of software-based art is an artwork made with a computer that needs a computer to function. This computer could, in fact, be replaced by any digital hardware capable of executing a program (a sequence of operations) (Butterfield, Kerr, and Ngondi 2016, 105). Tom Ensom defines software-based art more precisely, as "artworks where software is the primary mechanism in the realisation of the work and the primary material which the artist has chosen as a means of expression" (Ensom 2018, 18).

An artist's choice of software can be as deliberate or as contingent as any other material used to create contemporary art. The conservation scholars Engel and Philips recognise the uniqueness of an artist's source code, and maintain that this must be respected when treating the work (Engel and Phillips 2019, 186). They refer specifically to Shu Lea Cheang's web-based artwork *Brandon* (1998–1999), which does not use Cascading Stye Sheets (CSS) to define the layout of web pages. The World Wide Web Consortium introduced CSS in 1996 as a standard to facilitate the layout of web pages. "Although CSS was available when *Brandon* was created, its programmers decided against its use, possibly because the technology was nascent at the time or perhaps because styling uniformity across *Brandon* was not an aesthetic priority for the artist and her collaborators" (Engel and Phillips 2019, 185). Nick Montfort (Monfort 2008) provides another example of a specific programming style called obfuscated coding. He explains that lay programmers took pride in writing one-line BASIC programs to create unexpected outputs. The same outputs could be reached by code that was more descriptive and transparent about what it did, but this was not the idea behind obfuscated coding. In such cases, the conservator must therefore be able to recognise this coding style or practice in order to respect it. I treat software as an artistic material here based on my experience that conservators are usually unfamiliar with it, and computer scientists prefer to write their own code rather than trying to understand somebody else's. As such, it is important that conservation theory acknowledges software as an artistic material on a par with paint or plaster.

Ensom mentions liminal materiality as a property of software-based art. Software "simultaneously occupies multiple material states, without definitively belonging to any of them" (Ensom 2018, 59). This liminal materiality is also mentioned by Annet Dekker as a form of ambiguity that allows the artist to play with the meaning of an artwork. To illustrate the point, Dekker mentions *http://wwwwwwwww.jodi.org/* (1993) (Dekker 2014, 14), a web-based artwork by the artist collective Jodi. Upon opening the website, one can only see ASCII signs in a seemingly random order with no obvious meaning. However, viewing the source code in the web browser reveals it to visually form a hydrogen bomb. The artwork

plays with user expectations: the artists have inverted the typical arrangement which would see an image on the web browser surface and the ASCII characters in the concealed source code. This is only possible due to an interplay between the piece's different layers: the website's source code (its HTML code), the web browser (which interprets the HTML code), and internet protocols (HTTP and lower-level protocols that allow for information exchange between user and server). As such, the work's materiality cannot be pinned down to single components. The artists MTAA have illustrated this dynamic with an image showing two computers connected via a cable, with the connection itself labelled "The art happens here" (Figure 6). This kind of liminal materiality is a challenge for conservation, which could be compared to the translation of a poem whose verses convey several meanings simultaneously.

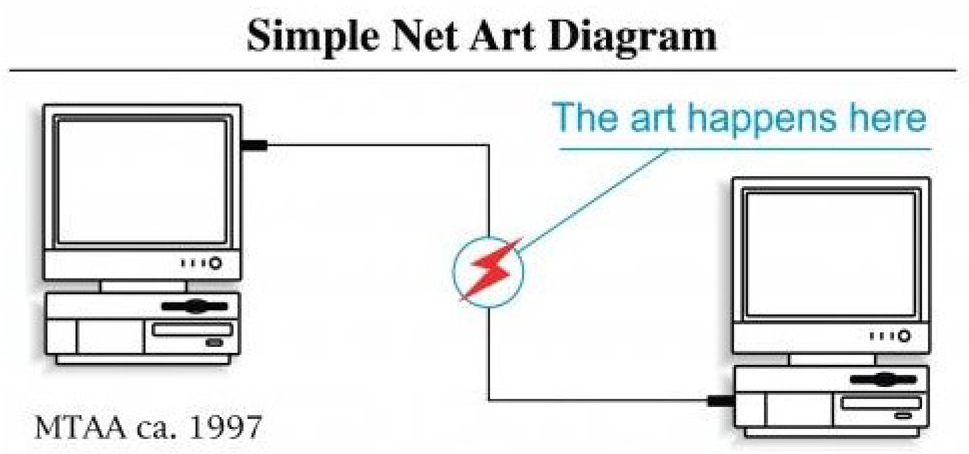


Figure 6: MTAA, "The art happens here". Source: (Sarff and Whidden 1997)) (quoted in (M. Connor et al. 2019, 4)

Nathalie Heinich (Heinich 2014, 34) has described contemporary art as a game with boundaries. Such art – precisely due to the choice of software as a material – transgresses the boundaries of classical or modern art. Software allows artists to break away from static physical objects and create dynamic objects or virtual spaces. Liminal materiality is, therefore, another playful way to transgress boundaries. Internet-based art then crosses a further boundary because access to it is not limited to the space of the museum; it is open to anyone.

As one of my case studies is an internet-based artwork, it is worth mentioning Dekker's definition of net art: art that is networked, and independent from its underlying technology (Dekker 2014, 28). As my research focuses on software-based art, however, I restrict this definition to the medium of the internet and use Baumgärtel's definition: "net art

deals with the specific conditions the internet offers (…). It plays with the protocols of the Internet." (Baumgärtel 2001, 24). However, as the term is today sometimes used to refer to net art made in the 1990s, I prefer the terms internet-based art and internet art. Furthermore, I sometimes use the term web-based art for artworks that are located on the World Wide Web, namely on the HTTP protocol, which is more specific than internet-based.

Dekker places internet art alongside the practices of contemporary art as defined by Heinich, who states that "the artwork is no longer exclusively the actual object proposed by the artist, but rather, the whole set of operations, actions, interpretations, etc., brought about by this proposition" (Heinich 2014, 35). Heinich later states that for contemporary art, the artwork not only consists of the object proposed by the artist, "but also the whole context of the proposition" (Heinich 2014, 42). As context is often a key part of the work, conservators face the problem of having to set boundaries for their conservation measures. This is particularly the case for networked artworks (artworks that are interconnected with other users or computers) as opposed to self-contained artworks that have no external software or data depedencies.[4] Caitlin Jones highlights the fact that some internet-based artworks are hard to understand without additional context (C. Jones 2006). This attention to context is crucial for conservation, for both contemporary and software-based art, and needs to be considered when defining the object of preservation.

Dekker chose as a case study for her dissertation the internet-based artwork *www.mouchette.org* by Martine Neddam, which is in a permanent state of change as the artist continues to work on and adapt it. In contrast, I focus on software-based artworks that are completed projects. By completed, I mean artworks that the artists have stopped working on, and whose software projects have ceased to be developed further. Software-based artworks that rely on further or participative code development need different skills and approaches, and as such do not feature in my research.

Because software requires the execution of instructions, it always has a performative and processual aspect, even if the artwork has stopped evolving from an artistic point of view. Processual and performative characteristics are at the heart of the conservation dilemmas described by (Vall 2017). In the past, conservators were trained to conserve the

---

4 The Variable Media Network defines behaviours of media art as such: a *contained* artwork is an artwork that is contained within its material and/or enclosed within a protective framework (Berkeley Art Museum / Pacific Film Archives Berkeley et al. 2001). It has no connections to its environment. A *networked* artwork is "designed to be viewed on an electronic communication system, whether a Local Area Network or the Internet" (Berkeley Art Museum / Pacific Film Archives Berkeley et al. 2001). It is interconnected with other users, computers, or servers.

material of an artwork, whilst processual or performative artworks may instead require the preservation of their concept or score.

Applying Nelson Goodman's concept of autographic and allographic art to time-based media art, Pip Laurenson offers a description of this performative and processual aspect (Laurenson 2006, 4). Autographic art, such as painting, depends on the material of the object, which cannot be replaced because it is uniquely applied or produced. It can be forged, because the fake can be differentiated from the original based on the material used (Goodman 1976, 113). Allographic art, such as a piece of music, is created in two stages: first a score is created, which is then interpreted by a musician. Allographic art cannot be forged, as new manifestations are simply new interpretations of the score (Goodman 1976, 114). For time-based media artworks – which, according to Laurenson, have both autographic and allographic aspects (Laurenson 2006, 10–11) – it is often difficult to know where exactly a conservator's focus should fall. This thesis will not resolve this dilemma, but it will provide a more differentiated range of conservation strategies in order to handle it better.

As software is a crucial component of software-based artworks, the following section is a discussion of its characteristics that are most relevant to preservation.

2.2 Conceptualisation of Software for Preservation

Francesco Amoretti and Mauro Santaniello define software as an "unambiguous sequence of instructions that allows a machine to elaborate information" (Amoretti and Santaniello 2009, 229). For this dissertation, I assume that the machine in question's technology is digital. Amoretti and Santaniello's definition does not mention data as part of software. However, classing something as data or code depends on one's point of view. Source code can be data from a compiler's perspective, or instructions from a programming perspective (Suber 1988, 14/22). Software, though the antonym of hardware, is data stored on computer hardware. Most of this data is modifiable and contains instructions, configurations, and input data for the hardware to execute computations. In other words, software is the information stored on a computer (or computer network) required to run the artwork. It consists of the software artefact which is specific to the artwork and its software environment of ancillary software, libraries, device drivers, and configurations. In practice, the difference between artefact and environment is often unclear, and there will be "fuzzy"

software elements that can be assigned to both software artefact and software environment, which has consequences for conservation.

For the preservation of software-based art, the difference between software stored in a repository and software installed and configured on a computer is crucial. To install software in such a way that the artwork behaves as intended not only requires all of the correct dependencies (the software environment), but also a knowledge of how they are to be installed and configured. This implicit knowledge can include many details whose importance is not immediately obvious – even with a proper understanding of the artwork – but that could lead to the artwork malfunctioning if configured incorrectly.

Over the course of the following paragraphs, I will further describe software for conservation. Here, the computer scientist Leon Osterweil's characterisation of software is also relevant for conservation. According to him, software is non-physical, but is often designed to control physical ("tangible") processes (Osterweil 2008, 266). As the hardware that executes these physical processes will eventually need replacement, the compatibility between software and hardware will always be an issue in the long term.

Another consequence of the interplay between software and hardware is software performance. The software performance model of the National Archives of Australia (NAA) states that the result of executing software on a piece of hardware is a performance (A. Wilson 2005, 21). The performativity of software therefore poses a challenge for conservation as it can be difficult to achieve the same performance or behaviour with different hardware. This may result, for instance, from a different computer clock rate or – more generally – a different runtime (concerning the technical environment, including both hardware and ancillary software).

Osterweil further describes software as having a hierarchical structure, as being "expected to require modification / evolution", and as comprising many interconnections (Osterweil 2008, 266). An artwork component such as software that is constantly evolving and subject to change is necessarily a challenge for any form of conservation that intends to minimise change. Due to the increased complexity of the software, these interconnections between different software components will therefore reinforce this challenge. Indeed, software complexity is an important consideration in the preservation of software-based art. It consists of structural complexity, as defined by Tom Ensom (Ensom 2018, 60), and source code complexity. Structural complexity refers to the many relationships between source code files and components such as software libraries, drivers, hardware, and external dependencies, such as web services. Source code complexity, meanwhile, is the complexity

within the source code. There are several metrics that describe it, such as cyclomatic complexity (McCabe 1976, 308), and the complexity of program evolution (Belady and M.M. Lehman 1985, 336). Whilst structural complexity is always a feature of software-based art, source code complexity is not necessarily inherent to it, but it is a risk, as software complexity is increased with each update and migration.

Opacity is another software characteristic produced by the layered and complex structure of software (Ensom 2018, 60). The more software libraries and layers are involved and the less documented a software is, the more opaque it becomes, and the more difficult it is to predict its behaviour. The ability to understand what a piece of software is doing has implications not only for its preservation, but also for the way it is perceived. Processes that occur in the background are invisible to the user, with even massive changes to the software remaining potentially hidden from the user interface. Special efforts are required to make this kind of change visible, and whilst such visibility may not be desirable, forgetting that a change took place may constitute a risk if it is not documented appropriately.

Ensom points to multiplicity as another of software's characteristics (Ensom 2018, 60). Because software can be duplicated so easily, different versions can exist simultaneously. In the case of the acquisition of software-based artworks, institutions usually acquire software as a single version. However, the way in which software comes into being often resembles the multiple branches of a tree growing outwards. Moreover, conservation processes themselves often add new software versions to that initially acquired. This proliferation of versions is, therefore, a challenge for conservation management, as these versions and instantiations are difficult to integrate into art museums' existing collection management systems. Questions such as, for example, under which conditions a new version would be registered need addressing (indeed, what constitutes a new version?). It is also worth noting that a software's complexity also usually increases as a result of changes applied, unless particular attention is paid to minimising such complexity (Belady and M.M. Lehman 1985, 253).

Software has functionality. It executes instructions in real-time and can react to user inputs, as opposed to other media such as film and video which are merely played back, resulting in the same output every time (Ensom 2018, 60). In that sense, software-based art can have a use. This use, or function, is often a significant property of the artwork, and should therefore be preserved.

What is missing in this list of properties with regard to conservation strategies is the fact that computer systems are based on abstraction. Ensom offers a description of the

principle of abstraction (Ensom 2018, 34), but does not include it in his list of the characteristics of software-based art. Instead, he uses the more general term structural complexity. As identifying the layers of abstraction is often crucial for the preservation of software-based art, I propose the term abstraction as another important characteristic of software. Abstraction serves to reduce complexity and is a design principle within software engineering (see Ghezzi, Jazayeri, and Mandrioli 2003, 49] and Fielding 2000, 5]). Layers of abstraction are built on top of, and depend on, each other. Software architecture abstraction is a key type of abstraction, where the runtime elements of a software system are abstracted. Its aim is to make applications more independent from platforms and hardware, and this is also a goal of preservation. It also facilitates communication within, for instance, computer networks, without the programmer or user having to engage with the technical details of the underlying technology. Language abstraction frees the programmer from potentially convoluted machine coding. High-level languages are closer to the task that a programmer must program than machine code. It is then the job of compilers or interpreters to translate this high-level code into machine code. Procedural abstraction is common within computer programs. It enables the programmer to refer to procedures or functions by their name, and means they do not have to write out the entire code of these procedures every time they use them; it is enough to refer to them. This method of procedural abstraction increases efficiency, lessens complexity, and facilitates collaboration between programmers. It also makes the code more readable, which is an advantage for conservators. However, procedural abstraction creates dependencies on external libraries which have to be consulted during the preservation process.

Abstraction is more efficient if used in combination with standards. Not only is computer hardware standardised to a certain extent; software is too. One example of standardised components – OpenGL for video cards and graphics software – facilitated the programming of game engines. Another example are the internet protocols that enable communication between internet nodes. In software engineering, the terms generalisation or generality are preferred to standardisation (see Ghezzi, Jazayeri, and Mandrioli 2003, 52], or Navrat and Filkorn 2005, 100]), and refer to the grouping of problems according to similar characteristics so as to find a common solution. Generalisation increases the ability to reuse software components, which is significant for the long-term preservation of software-based artworks. Abstraction and generalisation are characteristics of computer systems and, therefore, of software-based art.

To summarise, software usually has a software environment, runs on tangible hardware, is complex, hierarchical, modifiable, opaque, can multiply easily, is performative, has functionality, and is based on the principle of abstraction and generalisation. It is important to differentiate between installed software, with its implicit knowledge of dependencies and configurations, and stored software artefacts that do not include this information automatically. I will come back to these characteristics in the next chapter.

2.3 Work-Defining Properties, Authenticity, and Change in Conservation Theory and Digital Preservation

Due to the fragility of software-based artworks and the unavoidable introduction of changes when conserving them, the question of what is significant in an artwork must be raised. This section examines the ways in which conservation literature discusses the work-defining properties of an artwork, its changeability, and how these relate to the matter of its authenticity.

The variability of media art was novel for traditionally educated conservators at the end of the 1990s. This led to discussions and research carried out by the Variable Media Network[5] into the variability of artworks for conservation practice (Berkeley Art Museum / Pacific Film Archives Berkeley et al. 2001). Installation artworks are only perceivable when installed; otherwise they are simply pieces of equipment and digital files. Upon installation, artworks have to be adapted to the space and exhibition context. They are often flexible in terms of equipment used, spatial arrangements, sound volume, and other configurations which can result in very different presentations of the artwork. As Laurenson states, installations can be “thickly or thinly specified” (Laurenson 2006, 5). For thickly specified artworks, the artist provides detailed installation instructions. For thinly specified artworks, the artist leaves decisions about adapting the artwork to the exhibition space up to curators and conservators. To make its curatorial decisions transparent, the Guggenheim Museum introduced so-called Iteration Reports (Guggenheim Museum) drawn up by the conservators and technicians responsible for installing an artwork.

In 2014, Laurenson introduced the concept of work-defining properties for software-based artworks, a term that comes initially from the philosopher Stephen Davies (Laurenson

---

[5] The Variable Media Network consisted of US and Canadian contemporary art museums and foundations such as the Guggenheim Museum, New York, the Daniel Langlois Foundation for Art, Science, and Technology, Montreal, Franklin Furnace, New York, Berkeley Art Museum/Pacific Film Archives, Berkeley, Rhizom.org, New York, and the Walker Art Center, Minneapolis.

2014, 76). Laurenson developed a checklist comprising the areas of focus for software-based art which included technical dependencies, rules of engagement, and visitor experience as work-defining properties (Laurenson 2014, 92–94). Even with the aid of the checklist, however, it is not easy to describe the core of a software-based artwork, as this is subject to divergent interpretations.

In response to this, and to extend Laurenson's work on significant properties, Castriota proposes a concept based on Derrida's theory of "a structure without centre" (as described in Derrida's *Structure, Sign, and Play in the Discourse of the Human Sciences,* (Derrida 1967/1988 (reprint)). According to Castriota, an artwork's identity is not fixed but develops with every manifestation of the work, each iteration differing slightly from the last (Castriota 2019b, 116). The properties of these manifestations condense into a critical mass, the core of the artwork. Those outside of the core could be considered variable properties.

Castriota's model is based on a study of instantiations of artworks installed by different artists and institutions. The work-defining properties evolve, whilst the core of the artwork is permeable and can shift slowly (Castriota 2019b, 115). To adapt the work-defining properties to these slow changes, Castriota suggests that they be defined via a recursive process which is sensitive to considerations such as new instantiations of the work, art historical evaluations, and visitor reactions (Castriota 2019b, 171). This model works very well for software-based art with its multiple versions and simultaneous instantiations.

In the digital preservation field, archives and libraries use the term significant properties. An artwork's significant properties should "ensure the continued accessibility, usability, and meaning of the [digital] objects, and their capacity to be accepted as evidence of what they purport to record", and fall into five categories: content, context, appearance, structure, and behaviour (Grace 2009, 5). In contrast to the work-defining properties that provide a more or less complete definition of the artwork, these significant properties are those that are selected to be preserved for a specific community. Hence, significant properties are a subset of work-defining properties. This is confirmed by Rhiannon Bettivia's research into the significant properties of video games and the different needs of various communities such as researchers, game developers, and gamers (Bettivia 2016b, 29). For each of these groups, as Bettivia shows, a different set of significant properties applies. Therefore, significant properties have to be negotiated with these communities. Ensom also recognises this discrepancy between time-based media conservation and digital preservation. As he explains, art conservation usually privileges artist intent over user interests, whereas digital preservation focuses more on user requirements (Ensom 2018,

140). In art conservation, user requirements are secondary to the artist's intent and users tend not to be asked directly about their needs.

From now on, I will differentiate between *work-defining properties*, which encompass a complete description of the artwork, and *significant properties*, those properties selected for preservation (and therefore fewer in number).

Different interpretations of an artwork's meaning can also lead to the identification of other significant properties. One such example, for a small core of properties, are the different reinterpretations of Dan Graham's performance *Audience/Performer/Mirror* (1975 / performed 1977 in Amsterdam, see LI-MA n.d.a]) organised by LI-MA (LI-MA 2020). Gabriella Giannachi gives an account of these reinterpretations, stating that "each new work [reinterpretation] might therefore highlight, and so help to preserve, a different aspect or iteration of the 'original'" (Giannachi 2022b, 18). Interpretation takes place not only when reinterpreting an artwork, but also when conserving it with conservation strategies. The interpretative process that occurs when identifying an artwork's significant properties is well known within the field of art conservation. According to such one conservator, Barbara Appelbaum, "Objects have different meanings to different people. The differences are derived from culture, individual personality, social class, and the personal connection of the owner to the object in question" (Appelbaum 2007, xix).

Especially in cases where the core of an artwork's significant properties is small compared to the number of properties outside of its core, it is important to keep track of the versions of the work to understand its evolution and be able to return to earlier – or better – interpretations. Ensom calls this linking of versions, of the conservation decisions behind them, and of the socio-technical context needed to create narratives "the continuum approach to artwork biography" (Ensom 2018, 204). This continuum approach is relevant not only for conceptual works that need reinterpretation, but also for installations and performative or processual artworks. It strengthens the "tokening link" between the actual manifestation and the platonic form of the artwork to create an "authentic" version (Castriota 2019b, 161). Viviane Van Saaze observes that installation artworks can exist in *multiple instantiations* in parallel, each one with slightly different properties. This is possible because some consist of industrially fabricated, replaceable equipment, and of media that can be duplicated. These characteristics are also typical of software-based artworks. Van Saaze suggests describing the continued life of an artwork so as to "provide room to account for alterations (…)" (Saaze 2013, 106), and to accept this multiplicity as part of the artwork's authenticity (Saaze 2013, 107). Her concept of continuity frees

artworks from having to return to the state they were in when acquired. However, she advocates for slow changes, so that "the agreement of sameness" is never threatened (Saaze 2013, 105).

Van Saaze's concept of continuity, Ensom's continuum approach, and Castriota's concept of a structure with a permeable core all conceptualise the changes an artwork may undergo in relation to its identity. Although van Saaze notes the influence of institutional policies on the trajectory of Nam June Paik's installation *One Candle* (Saaze 2013, 107), she does not conceptualise it in her concept of continuity, and neither do Ensom and Castriota. Appelbaum acknowledges the institutional influence more explicitly. For her, "Institutions that own objects give them meanings based on their missions and programs" (Appelbaum 2007, xix). These meanings "affect the desirable goal of treatment" (Appelbaum 2007, xix).

To conclude, not only are some work-defining properties variable, but the preservation of software-based artworks is also subject to a process of interpretation. The identification of significant properties is influenced by conservators, curators, and institutional policies and interests. Conservation theory should, therefore, acknowledge institutional differences in the interpretation and selection of an artwork's significant properties.

Furthermore, there is no objective authority that identifies both its work-defining properties and the subset of significant properties. Instead, these are the result of a discursive practice as well as the process of progressively changing instantiations, and are often renegotiated when preservation strategies need to be chosen. The fact that the significant properties can be renegotiated during each preservation cycle makes it hard to select appropriate long-term solutions and increases the effort required for documentation. This must be taken into account when thinking about long-term preservation.

2.4 Conservation Strategies

This section examines different definitions of conservation strategies in conservation literature and how these strategies were applied in art museums and digital archives. I check whether the definitions are consistent in both art conservation and digital preservation and whether adaptations are necessary for software-based art.

Conservation Strategies in Art Museums and Digital Archives

Not only do art museums and digital archives preserve objects with different characteristics, but they also use a slightly different terminology for this activity. In art museums, the term conservation is more popular, whereas for digital archives, the terms preservation or digital preservation are preferred. Conservation and preservation both cover the range of actions from preventive conservation, which changes the environment of an object but not the object directly, to conservation interventions, which change the object significantly, such as restoration. In this dissertation, however, I make no difference between the terms conservation and preservation. As software-based art intersects the fields of art conservation and digital preservation, I use these terms interchangeably.

A strategy is a combination of a long-term goal and the methods employed to achieve this goal under conditions of uncertainty (Barad 2018, 3). Uncertainty is a familiar feature of conservation practice. When a conservation decision has to be made concerning software-based artworks, it is not usually known which technology or material will prevail, and what social interactions with technology will look like in the future. Furthermore, due to the complexity of such artworks, and a scarcity of resources, it is rare for conservators to be acquainted with an artwork's every minute detail.

Preservation measures or preservation interventions are the steps taken to implement a preservation strategy, but can also be undertaken as single events outside of a preservation strategy.

Interestingly, conservation strategies are primarily discussed in relation to time-based media conservation and digital preservation, where distinct strategies such as migration and emulation are identified and explained (see, for instance, the Variabe Media Network Berkeley Art Museum / Pacific Film Archives Berkeley et al. 2001] and Strodl et al. 2007, 29]). In other disciplines, such as painting or built heritage conservation, conservation strategies are generally not categorised or named as strategies.

As I am dealing with software-based art, I will analyse both digital preservation strategies applied in archives and libraries, and conservation strategies used in the conservation of time-based media art. As such, the following sections offer a discussion of the most significant research about and applications of these strategies in both fields.

Storage

Storage can apply to both hardware and software. I mention hardware storage here because software-based artworks often depend on their hardware much more than digital

objects within digital archives. This is why digital archives often have no hardware storage, except for specific cases such as video games. One could consider hardware storage to be the traditional approach of museums to the conservation of software-based art. Here the definition provided by the Variable Media Network remains relevant, as it encompasses both the physical storage of hardware and the archiving of digital files without changing them (Berkeley Art Museum / Pacific Film Archives Berkeley et al. 2001], select “terms” and “strategies”). Software and hardware storage are a basic conservation strategy carried out as part of the acquisition process, and as such are a prerequisite for further conservation strategies.

For software-based art, it often makes sense to store the hardware with the software installed on it. This has three distinct advantages: firstly, it guarantees that the software/hardware package stays together, remains compatible, and that the software and hardware configurations are preserved. Secondly, one can assume that no software library or other software components are overlooked (this is only valid for contained artworks without internet access). Thirdly, the software/hardware package can serve as a reference for the behaviour of the work when conservation measures are planned. In the case of artworks with dedicated equipment that has a bearing on the meaning of the work, hardware storage is particularly important. Hardware storage must be accompanied by software storage to ensure that the digital files can be accessed.

In digital archives, the storage of digital objects such as images, videos, text documents, and audio files is routine. This storage strategy is also called bit preservation, as the digital objects are preserved on the bit level. Bit preservation ensures that file content and format do not change, but it does not guarantee that the content will remain usable and readable (DPC 2019). The first three NDSA levels of digital preservation offer recommendations on how digital repositories can achieve this: by making multiple copies, keeping them at different locations, and checking their integrity at regular intervals, amongst other measures (The National Digital Stewardship Alliance 2019). A growing number of art museums such as Tate and MoMA have begun to apply bit preservation to save the digital components of their artworks. Based on their experience, they have produced an online guideline for other museums (MoMA et al. n.d.).

Such digital archives are also known as digital repositories. However, a digital archive must not be confused with version control systems such as Git (Git community n.d.), where the term repository refers to a collection of source code files. Version control

systems are not meant for the secure long-term storage of digital objects. Their main purpose is software development and collaboration.

Few digital archives store software objects. Software storage is mainly practised in scientific research communities, as in the case of Software Heritage's software archive (Software Heritage Team n.d.) or Zenodo (Zenodo n.d.). These repositories store software as source code. If the code is meant to be reused and further developed, it is usually transferred to a version control system. Software developers and researchers do not usually store software environments, such as the operating system, in the version control system. The purpose of software storage as practised by research communities is to share and reuse code as well as to ensure the reproducibility of computations. The journal ReScience C (Rescience C n.d.) is dedicated to the reproduction of scientific computations to improve research quality and promote open research practices.

Museums store the software for their software-based artworks, but in a way that is less unified than research communities. The software is stored at acquisition, when preparing for exhibitions and loans, and when carrying out conservation measures. As such, the software is stored not only as source code, but also as disk images and executables. This includes data such as text, images, sound and video, software configurations, metadata, software documentation, and software libraries.

Software-based artworks often change. This is particularly true of artworks that are networked and require user interactions. The relationship between the living object and the digital repository which stores the objects for long-term preservation must be established for each artwork. In other words, whether, when, and how a snapshot of the software-based artwork is taken upon its entry into a digital repository has to be decided on an artwork-by-artwork basis.

Migration

The Variable Media Network, founded by the Guggenheim Museum (New York) and the Daniel Langlois Foundation in 2001, shaped migration strategy primarily around the question of equipment, as video preservation and installations were the subject of lively discussion in contemporary art conservation at that moment:

> To migrate a work involves upgrading equipment and source material. The obsolete fluorescent bulbs of Flavin's light installation could be upgraded to fluorescent or halogen lights of comparable hue and brightness. The major disadvantage of

> migration is the original appearance of the work will probably change in its new medium. Even if state-of-the-art fixtures cast similar light to Flavin's originals, the actual fixtures are likely to look different. (Berkeley Art Museum / Pacific Film Archives Berkeley et al. 2001, select “terms” and “strategies”)

Upgrading equipment means replacing hardware with newer models. This hardware migration implies adapting the software involved. The Digital Preservation Handbook relevant to the field of archives and libraries defines migration as

> a means of overcoming technological obsolescence by transferring digital resources from one hardware / software generation to the next. The purpose of migration is to preserve the intellectual content of digital objects and to retain the ability for clients to retrieve, display, and otherwise use them in the face of constantly changing technology. (Digital Preservation Coalition 2015)

This definition focuses on the preservation of information and the usability of the digital object. It encompasses file and software migration. In digital preservation, migration is mainly applied for format migrations of single files, and is the most common method for ensuring that digital documents remain accessible. Every few years, document formats have to be updated to current standards in order to stay accessible via up-to-date software. As the backward compatibility of many applications only covers a few previous generations for input file formats, file format migration is inevitable if current access software is to be used. In art museums, file format migrations are mainly conducted to generate access copies for digital videos.

In both digital preservation and art conservation, it is still less common for software to be the object of migration. If software is migrated to a new environment, not only is its format updated, but so are its source code and the resulting executables. Such changes are often necessary as a consequence of updated operating systems, or of updated interfaces and APIs.

For the purposes of this dissertation, my focus on software migration will define it as follows: software migration means the transfer of a software artefact to an updated or different software environment in order to sustain its functionality. This transfer can be related to new hardware, but this is not mandatory. It may include manual adaptations of the software to the new environment or software updates distributed by software developers, and it may require either a recompilation of the source code or the installation of additional

software components such as libraries and drivers. The software artefact itself is changed and adapted to the new environment (JISC 2008). I consider the reprogramming of a software's functionality with a different programming language to be a separate preservation strategy.

Restructuring code to improve its transparency without changing its behaviour is called refactoring.[6] This form of migration is sometimes mentioned in the field of software engineering and maintenance. It aims to reduce complexity and maintenance, and to improve the readability of the code. Refactoring bears the risk of introducing errors to the code and affecting other parts of code that depend on the changed part. It may also intervene in the programming style of the artist and/or programmer.

Through migration, both file-based objects and software-based objects undergo changes that are sometimes difficult to foresee. Questions of authenticity emerge, especially where the object's look and feel – not simply its informational content – are important. Furthermore, migration usually has to be repeated, as the environment of the software artefact changes over time. Repeated migrations will undoubtedly change the software artefact, and the significant properties may become compromised in the long term. Also to be aware of is the likelihood that conservators and curators slowly adapt the artwork's significant properties to newer migration versions. Furthermore, the software complexity (Belady and M.M. Lehman 1985, 250) and the risk of errors can increase. Checking and documenting the artwork's significant properties after each migration is, therefore, essential.

I will now offer a detailed discussion of the migration of software-based artworks in museums, about which surprisingly little has been written in the conservation literature. An early mention of a software-based artwork is Gary Hill's *Between Cinema and a Hard Place* (1991) in an article by the conservator Pip Laurenson (Laurenson 2001). This is one of the first texts to describe the software migration or reconstruction of an artwork in the museum context. Hill's artwork is an installation based on customised CRT monitors, video laser disks, and a computer program controlling audio and video sequences. The software was designed by Dave Jones, and the program based on it was written by Gary Hill (Laurenson 2001, 264). The artwork questions and expands the video art genre by, on the one hand, introducing software that interrupts linear playback, and, on the other, laying bare the electronic innards of the monitors. The conservators were able to describe all of the

---

6 "Refactoring encompasses any changes to the system that leaves its behavior unchanged but enhances some nonfunctional quality (i.e. simplicity, flexibility, understandability, or performance)" (Bowers et al. 2002, 108).

software's functions. The argument for migrating or reconstructing the software based on this documentation was that computer and software would not be visible:

> The laser disc players, discs, audio equipment, and computer control system are the functional elements out of view, and the conservation strategy is different from the approach to functional elements that are visible. If the technology fails and these elements become obsolete it would be acceptable to the artist to substitute these components with an entirely new technology but only if their function were the same. (Laurenson 2001, 263)

In 2010, conservators working on the "Digital art conservation" project at the ZKM Karlsruhe conducted several case studies in which artworks were migrated in order to preserve them. For each artwork, conservators and art historians conducted interviews with the artists(s) and researched each piece's art historical background thoroughly. Often, the migrations were carried out by the artists themselves, as for instance in the case of Marc Lee's *TV Bot 1.0* (2004) to *TV Bot 2.0* (2010), discussed in Chapter 1. Interestingly, the conservator called the migration from *TV Bot 1.0* to *2.0* a reinterpretation as the artist also adapted the website's graphic design (Gassert 2013, 395). Let us recall here that the artist has since "migrated" *TV Bot* a third time, due to the obsolescence of Flash-based streams in modern web browsers. I write "migrated" between quotation marks as, in fact, as opposed to migrating the software, the artist recoded everything from scratch.[7] Through the migration from *1.0* to *2.0* and the reinterpretation that resulted in *3.0*, *TV Bot* has changed significantly over its lifespan. It will be interesting to see how the artist further develops the piece now that Twitter has become X and started to restrict access to the Twitter API. Such repetition of migrations has not been researched until now. However, migrations often have to be repeated, and this leads to the question of what impact this has on an artwork's authenticity.

Another important contribution here are two papers about the restoration of Shu Lea Cheang's internet-based artwork *Brandon* (1998-1999) (Engel and Phillips 2019, 2018] and Engel et al. 2018]). *Brandon* (Cheang since 1998) tells the story of a trans man who was raped and murdered and questions society's attitude towards otherness, as well as exploring perspectives on both the potentials and limitations afforded by one's online persona regarding gender. More than just a website*, Brandon* comprised installations and performances. The artwork was commissioned by the Guggenheim Museum (Phillips et al.

---

7 Interview with the artist, January 11, 2021 (Lee 2021b).

2017) and restored between 2016 and 2017. As the Guggenheim wanted the user to be able to access *Brandon* via a current web browser, they opted for a migration of code instead of browser emulation. Engel and Phillips describe the endeavours to keep code interventions to a minimum. They introduced the term code resituation, which “aims to preserve the original artist’s code, or significant parts of it, while adding restoration code to reanimate defunct code to full functionality” (Engel and Phillips 2018, 2). In the case of *Brandon*, rather than adding further code within the partly dysfunctional script, an altogether new script was created without having to deactivate the original one. To document the changes applied, the conservators not only used a version control system, but also commented on their changes within the scripts. Hence, the migration that they conducted was precision manual work, not an automated updating from one version to the next. The migration restored several of the website’s functionalities, and as the migration was done very carefully, it preserved the website’s look and feel.[8]

To conclude, there are different types of migration. They always involve a change to the artwork at the level of code, and sometimes also at a behavioural level. Migrations involve a risk that the artwork’s significant properties will be gradually changed, snowballing into larger changes after several migrations. Given that migrations must be repeated regularly, one can ask whether migration is a sustainable conservation strategy and in which cases it should be applied.

Emulation

Unlike migration, which changes the digital object itself, emulation does not change the digital object, but focuses instead on the environment in which the object was created (JISC 2008). The principle of emulation is to protect the software artefact from change by encapsulating its software environment within an additional layer. This layer isolates the software environment from the new hardware environment and serves as a bridge to the new hardware.

For the purposes of this dissertation, I define emulation as a software that is capable of interpreting the machine language commands from a specific type of Computer Processing Unit (CPU) or computer architecture. The emulator translates these commands in real-time. In practice, emulators represent a complete computer system, and typically include a sound card and graphics or network adapters. Smith and Nair call this a whole

[8] Surmised from (Engel et al. 2018). The former website of 1998/1999 is no longer accessible.

system virtual machine (Smith and Nair 2005, 20), whereas for Guttenbrunner and Rauber it constitutes a full hardware emulation (Guttenbrunner and Rauber 2012, 161).

The real-time translation of an emulator requires high levels of hardware performance. If performance is insufficient, installing a classic virtual machine may be an appropriate solution. Smith and Nair call this a classic system virtual machine (Smith and Nair 2005, 23). As opposed to emulation, the guest system here has direct access to the host computer's CPU, and the system commands do not have to be translated from guest to host (see, for instance Guttenbrunner and Rauber 2012, 162]). This is only possible if the computer architecture of both guest and host is compatible.

It is worth mentioning here that, in both the literature and internet blogs and discussion boards, the terms emulation and virtualisation are not always used in the way that I define them here. Virtualisation, in particular, is often used in a more general way to encapsulate all types of virtual machines, including emulators. Hence my use of the term classic virtual machine, or classic virtualisation, for the specific type of virtual machine which is based on the abstraction of the same Instruction Set Architecture (ISA) as the host.

In 1995, Jeff Rothenberg published his landmark paper "Ensuring the Longevity of Digital Information" (Rothenberg 1998) in which he introduced digital emulation as a way to access digital documents in contrast to the method of migration already practised. This gave rise to further research amongst heritage organisations, such as the National Library of the Netherlands (The Hague, Holland) which commissioned a report into the uses of emulation to preserve digital publications (Rothenberg 2000). The TOTEM project suggested a metadata model for the description of software environments (Delve, Konstantelos, and Ciuffreda 2011), to automate the uploading and assembly of environments necessary to render digital objects. The KEEP project (KEEP 2009-2012) created portable emulators to enable users to access obsolete digital objects in reading room and exhibition situations. The BWFLA project at the University of Freiburg (2011-2013) designed a framework offering the user various emulators to create view paths for digital objects based on their native environment (Rechert et al. 2012). All of these emulation projects aimed to either test emulation or improve and develop its features, automation, and user-friendliness.

In 2018, the Software Preservation Network (The Software Preservation Network n.d.b) introduced the Emulation-as-a-Service infrastructure, EAASi, a cloud-based emulation platform for US-American university archives and libraries. It builds on the platform developed by the BWFLA project, and enables the sharing of software

environments and emulators to render archival assets. Recreating obsolete software environments is often time-consuming. Sharing software environments, however, has a scaling effect that can reduce the conservation effort significantly. Gradually, the EAASi platform became increasingly user-friendly, more automated, and better documented (The Software Preservation Network 2019, ongoing). In Europe, the Netherlands Institute of Sound and Vision tested the EAASi platform and provided positive feedback, but archives remain reluctant to apply emulation for the preservation of software and complex digital objects. Copyright issues and the effort needed to build suitable environments for their digital objects might be amongst the reasons for the slow uptake of emulation.

To my knowledge, the first time that art institutions applied emulation was for the exhibition *Seeing Double. Emulation in Theory and Practice* in 2004 at the Guggenheim Museum (Variable Media Network 2004). There, several digital artworks were presented as emulations or virtualisations alongside a version of the artwork without emulation. *Jet Set Willy Variations* (2002), a modification of a computer game by JODI, ran on a ZX Spectrum computer emulation, whilst *[phage]* (1998/2004) by Mary Flanagan, based on Macromedia Director, was presented with Windows Virtual PC on a Macintosh computer. In 2011, the ZKM Karlsruhe staged an exhibition of the digital artworks that it had preserved as part of its Conservation Digital Art project. There, the Quickbasic-based artworks *ABROLL3* (1994) and *RAHMEN4* (1992) by Franke were presented with the DOSbox emulator (Serexhe 2013).

Rhizome (Rhizome n.d.), an institution that collects and presents internet art, was probably the first institution to implement so-called web browser emulations that enabled users to access obsolete websites with obsolete web browsers via Emulation-as-a-Service developed by the BWFLA. Examples are listed in the book *The Art Happens Here* (M. Connor et al. 2019) and can be accessed online ([Hershman Leeson n.d.], [Lialina n.d.], and [Neddam n.d.]). These emulators are installed on remote (or cloud) computers that allow users to access the emulator without having to install it on their own computer. Instead of the term browser emulation, Rhizome refers to remote browsers in the book, as they are based on Linux containers[9] instead of full system emulators (Espenschied and Moulds 2019, 434). However, for many artworks, cloud-based emulators are too slow because each interaction and audio-visual change must be transmitted via an internet connection. That was one of the conclusions reached by Doren and Michaan who tested Emulation-as-a-

[9] A container is an operating system virtualisation (see also glossary).

Service framework for CD-ROMs (Doren and Michaan 2016, 12). They also criticised the visibility of the host computer's web browser window, as CD-ROMs are usually presented full screen.

In 2015/2016, the BWFLA developed a "portable emulator" called EMIL, a selection of emulators installed on a bootable USB thumb drive. When inserted into an exhibition computer, the previously selected emulator will boot up immediately and run in full-screen mode on this computer. Besides not having to rely on an internet connection and therefore performing better, this has the advantage of connecting specific (obsolete) peripherals to the computer where the emulator is running. This is particularly useful in an exhibition situation where the artwork is installed physically. Furthermore, it does not affect software installation on the exhibition computer whatsoever. Espenschied et al. used this emulation setup for Olia Lialina's internet-based artwork *My Boyfriend Came Back From The War* (1996) in different exhibitions over the course of 2016 (Espenschied et al. 2016, 39).[10] The EMIL thumb drive is also well suited to run (disk images of) obsolete CD-ROMs, as it automatically suggests the right emulator and can be used as a distribution media or exhibition copy instead of the obsolete CD-ROM itself. For museums, it can be a valuable distribution medium for the loan of digital artworks without hardware.

In 2014, Patricia Falcao et al. looked into classic virtualisation in the conservation department at Tate (Falcao, Ashe, and B. Jones 2014). Tate's Information Systems department was already using classic virtual machines at that time. Falcao et al.'s goal was to use this existing infrastructure to virtualise certain artworks and integrate this virtualisation into Tate's archival process. "General maintenance of the virtual machines would be carried out by the IS department, which maintains Tate's servers. (…) Conservation retains responsibility for testing the virtual machines at regular intervals and any major upgrades of the virtualisation platform" (Falcao, Ashe, and B. Jones 2014, 7).

To summarise, museums are familiar with and apply the strategy of emulation, but performance issues remain. However, emulation infrastructure is developing quickly. Sharing emulators and software environments is becoming increasingly straightforward. Furthermore, the emulation strategy also includes classic virtualisation. Hence, other types of virtualisation should be explored. The sustainability of emulation as a preservation

---

[10] Displayed at the exhibition "Electronic Superhighways (2016-1966)" at the Whitechapel Gallery and at the House of Electronic Arts in Basel, both 2016.

strategy – in terms of dependency on the emulator – remains to be seen, as does when it should be applied.

The Conservation Strategies of the Variable Media Network

In 2001, the Guggenheim Museum, the Daniel Langlois Foundation and other art institutions founded the Variable Media Network, to research conservation strategies for time-based media and installation art (Variable Media Network n.d.). Their definitions of conservation strategies have proved influential in the field of time-based media conservation. They helped conservators to move on from the idea of a static object, and gave them a conceptual base to do this. As conservators still refer to these definitions, it is well worth discussing them here.

At conferences such as the 1997 "Modern Art: Who Cares?" symposium in Amsterdam, conservators expressed their concerns regarding the conservation and documentation of variable artworks, and the paradox of preserving variability. The Variable Media Network addressed these concerns by suggesting four strategies that could be applied to variable artworks: storage, migration, emulation, and reinterpretation, and used case studies and an exhibition[11] to illustrate them. The case studies not only included time-based media artworks such as *[phage]* (1998) by Mary Flanagan or *Color Panel v1.0.1* (1999) by John F. Sion Jr., but also installations such as *Untitled (Public Opinion)* (1991) by Félix González-Torres and performances such as *Site* (1964) by Robert Morris.

These approaches to conservation were also applied in the Inside Installations project (Scholte and Wharton 2011) and tested on different case studies therein. One such example was the emulation of *Albert's Ark* (1990) by Bill Spinhoven (Wijers 2011). The core of the work, a piece of electronic hardware custom-built by the artist, had disappeared. In 2007, it was decided to replace this piece with a standard PC, and to recreate its behaviour with Visual C# based on analysis of a documentation video. This strategy was termed emulation, following the Variable Media Network's definition:

> To emulate a work is to devise a way of imitating the original look of the piece by completely different means. Emulating a Flavin fluorescent light installation would require custom-building fluorescent bulbs that produce the same light as and

[11] *Seeing Double. Emulation in Theory and Practice* (Variable Media Network 2004), s. previous section.

resemble the physical appearance of the original bulbs. (Variable Media Network n.d., see “terms”, “strategies”)

However, the computer emulation of *Albert’s Ark* (1990) does not constitute emulation as defined in the field of digital preservation: it is not based on a translation of the electronic hardware’s instruction set as built by the artist. Rather, it is a reengineering based on a documentation of the work’s behaviour. The term emulation can, therefore, be confusing in relation to the conservation of software-based art.

The storage and migration strategies of the Variable Media Network have already been discussed in previous sections and need no further explanation. They focused on equipment as, at that time, video preservation and installations were prevalent, but did also include software (source code). Besides these strategies, the Variable Media Network introduced reinterpretation, which it described as the most radical of the four strategies. In contrast to the other conservation strategies, reinterpretation is not material-based, and does not try to achieve the same appearance, but sets out to preserve the concept or idea of the work. The authors describe the reinterpretation strategy with reference to Dan Flavin’s light art: “To reinterpret a Flavin light installation would mean to ask what contemporary medium would have the metaphoric value of fluorescent light in the 1960s.” (Variable Media Network n.d., see “terms”, “strategies”). It is usually applied when the physical part of the artwork cannot be preserved or replaced.

To summarise, the Variable Media Network uses the same terms for conservation strategies as digital preservation, but with a greater focus on equipment. The strategy of reinterpretation was added to overcome technical limitations and enable current implementations. The Variable Media Network’s approach to emulation is not the same as in the field of digital preservation. To avoid confusion, therefore, these conservation strategies should be (partly) redefined.

Other Conservation Strategies

In addition to the strategies outlined above, a number of others exist. The Software Sustainability Institute focuses on the preservation of software used to produce scientific research results, and has developed its own approach to conservation strategies (Software Sustainability Institute 2011). In addition to hardware preservation, migration, and emulation, they cite cultivation and hibernation as preservation strategies. Cultivation means allowing other developers to reuse relevant code to produce new software or continue

current software, sharing the code and curating the development community. This is more than just a migration, as developers can add new features; the owner of the code no longer has full control over its development. In the domain of software-based art conservation, this may apply to projects with a participatory element where the artist wants their software to be further developed as an artistic strategy. In contemporary art, meanwhile, this would apply to processual artworks that have to be preserved according to the processual paradigm discussed by Renée van de Vall (Vall 2017, 84). The artist and/or museum would have to actively guide the development of the software-based artwork in such cases.

The Daniel Langlois Foundation's DOCAM project introduced the strategy of reconstruction besides those of storage, emulation, and migration examined above:

> Reconstruction consists of reproducing the behaviours and effects of a work using few or no components from the original piece. (...) The concept of reconstruction can also be applied to the installation and sculpture domains. In both, one seeks to reproduce the physical appearance of a work using new materials. The same techniques and assembly methods are not necessarily adopted, but the work's original shape and dimensions are nevertheless respected. (DOCAM 2010b)

Reconstruction, though mentioned neither in the digital preservation field nor by the Variable Media Network, should be recognised as a major conservation strategy. For artworks with lost parts such as the previously mentioned case of *Albert's Ark* (1990) by Bill Spinhoven (Wijers 2011) or external dependencies, reconstruction can be highly useful.

Due to the rapid obsolescence of internet technology and the ephemerality of (online) performances, Anne Laforet and Annet Dekker have pointed to documentation as a strategy for conserving digital art (Laforet 2009, 186) (Dekker 2014, 180). The use of archival material as a conservation strategy resonates with Hanna Hölling's theory that the archival material that documents and contextualises an artwork is itself a part of that artwork (H. B. Hölling 2015, 87). Recently, the Documenting Digital Art project, run by the University of Exeter in collaboration with LI-MA, showed how the documentation of complex, performative digital artworks can itself become part of the artwork, and highlighted the importance of documenting the visitor experience (Giannachi 2022a, 141). However, art museums have never embraced documentation as a fully-fledged conservation strategy for time-based media art, although it has been applied by artists: many performances in the twentieth century were recorded on video and now replace the live act. For such projects, the documentation can become something of a follow-up artwork after the

performance. Rhizome's net art anthology (Rhizome since 2016) provides contextual information for certain pieces of internet art, particularly older works that are no longer fully functional. The documentation of the live event *Telefonia* as a fluid archive online (Bosshard 2020) by one of the artists in collaboration with the House of Electronic Arts is just such an example; the online documentation of the defunct Facebook game *Naked on Pluto* as a Wiki archive (Barok and Boschat Thorez 2020) is another. These examples clearly illustrate that documentation can be a valid conservation strategy in itself. A more pressing question concerns the kind of documentation methods available, and whether they can increase an artwork's sustainability.

To conclude, reconstruction and documentation – although not mainstream – are valuable conservation practices and should be added to the list of strategies above.

Maintenance

In this thesis, I do not consider maintenance a conservation strategy, even though, in different contexts, the conservation of art may be considered a form of maintenance. According to the ISO standard (ISO/IEC 2022), maintenance consists of a series of actions to keep software operational, its purpose being to adapt the software to a changing environment. The ISO standard classes different types of maintenance, such as preventive, corrective, adaptive, additive, and perfective. Conservation ethics must always be taken into account when maintenance is carried out. Corrective maintenance is performed to correct software errors. However, if the error is considered part of the artwork, it should obviously be retained. Although adding features to a piece of software, such as in additive or perfective maintenance, is usually not considered a form of conservation, there may be cases where it is admissible, for instance when adding a feature such as screen or sensor recording for documentation purposes. Improvements made to software performance may result from perfective, adaptive (adaptation to a changing software environment), or preventive (prevention of malfunctioning) maintenance. Whether this improvement is desired or not depends on the significant properties of the artwork in question.

Updating an artwork's software can be considered as either a form of maintenance or migration. As the effects of an update are often unpredictable, I suggest the term migration for updates made directly to an artwork's software. I consider maintenance in those cases where the emulator or a reconstructed part of the artwork is updated. Other forms of maintenance include monitoring the functionality of the artwork and whether its software needs updating.

2.5 Decision Criteria for Conservation Interventions

Conservation strategies or measures are not chosen independently from an artwork's work-defining properties. Certain strategies support certain properties, and others do not. There is, therefore, some overlap between the criteria needed to determine these work-defining properties, and those required to select an appropriate conservation measure or strategy. Therefore, although this section focuses on the decision criteria for conservation strategies, it also includes those for determining an artwork's work-defining properties.

Such decision-making has been a topic of serious discussion since the beginning of the twentieth century. In 1903, Alois Riegl introduced a set of values by which the work-defining properties of a monument could be assessed (Riegl 1903). He wanted to move away from a paradigm in which a work's stylistic unity was the main driver for restoration, and stress the importance of historical importance. He suggested disentangling the various values assigned to monuments in order to identify the primary conflicts and make the decision-making process more transparent. Hence his system of values consisted of an artefact's age, historical value, artistic value, newness, and use value, which were to be weighed up against each other in the selection of conservation measures. Much later, Barbara Appelbaum adopted and refined Riegl's approach in her book on conservation treatment methodology (Appelbaum 2007, 89). She introduced additional values, such as research value, educational value, sentimental value, and monetary value.

These value systems articulate the dilemma within conservation when facing older artworks: whilst such objects have a historical value, they may no longer be able to convey their meaning due to age-related decay. This also applies to performative and processual art, but the rubrics of newness and age value are less relevant for their conservation. Such value systems are applicable to the conservation of software-based art, but certain terms need to be translated (how do newness and age value relate to software-based artworks?) and the variability, performativity, liminal materiality and multiplicity of the artwork have to be considered. The yellowing of a grey computer case, the size and clumsiness of an old computer keyboard, the now endearingly cuboid Apple Macintosh Classic with its small integrated monitor, the outdated design of a digital computer desktop with the familiar booting sound, the outdated design of a software's user interface; these can all cause nostalgic feelings that testify to the potential age value of both hardware and software. Whilst collectors of old computers and games value the age of such items, the question for

contemporary art curators is whether to accept that contemporary art can also have an age value.

The newness value, meanwhile, is often a challenge for software-based art. Such art is often based on new technology, and can surprise viewers when first shown. However, ten years later, the same artwork is liable to provoke very different reactions. So, whilst the artwork had a newness value, can this be reproduced, and does it have to be?

The use value of an artwork is even more difficult to describe, as artworks tend not to have a direct purpose. I would understand use value in terms of an artwork's functionality, as software-based artworks are usually executing processes. Use value, therefore, also indirectly accounts for the performative/processual nature of software-based art. Whilst the variability, liminal materiality, and multiplicity of such artworks are not covered by this value system, they do not interfere with it. This shows that Riegl's value system remains relevant for software-based art today, and that it can support the selection of significant properties and hence indirectly the choice of conservation strategy.

At the 1997 "Modern Art: Who Cares?" symposium, contemporary art conservators recognised the need for a common terminology and methodology within their discipline. In 1999, a working group of the INCCA (International Network for the Conservation of Contemporary Art) published a model to guide decision-making relating to the conservation of contemporary art in a book named after the conference. It suggested the use of flow charts as a procedural tool for selecting one out of several treatment options based on the difference between a work's intended meaning and its current condition (INCCA International Network for the Conservation of Contemporary Art 1999, 165). However, as Goodman points out (see section 2.1), the object to be preserved is not necessarily an autographic work, and the justification for choosing a particular treatment is not always material damage or decay. Furthermore, the idea that an artwork has one essential meaning is of course highly flawed, as this fails to recognise both the subjectivity of the viewer and the layered reality of all meaning, artistic or otherwise.

In 2020, the Cologne Institute of Conservation Sciences revised the INCCA's decision-making model (Cologne Institute of Conservation Sciences / TH Köln 2019, revised 2021). The revised model mentions not only conservation but presentation as a goal for intervention. This is highly relevant for web-based artworks, where presentation and conservation coincide. However, the new model does not include any long-term perspectives on artworks that go through many conservation cycles. Rather, it focuses on one conservation cycle, only hinting at a repetitive process. I would go further and say that

conservation decisions should incorporate a long-term perspective, so as to facilitate future conservation measures, and should begin with the assumption that there will always be more versions of a work to follow.

The new decision-making model lists the following decision criteria for conservation/restoration interventions: “aesthetic and artistic values, authenticity, historicity, functionality, artist’s opinion, relative importance of the artwork, financial limitations, legal aspects, technical limitations, restoration/conservation ethics” (Cologne Institute of Conservation Sciences / TH Köln 2019, revised 2021, 14). Some of these resonate with Riegl’s values, whilst others such as “functionality” and “technical limitations” recall the work-defining properties of software-based artworks as defined by Laurenson (2014, 92–94; see also section 2.3 of this dissertation). Of importance here is that the revised decision-making model mentions financial limitations as a decision criterion, which I am yet to see discussed in the conservation literature, despite its significant impact on the choice of conservation strategies. As I judge this factor to be fundamental within the conservation decision process, I will build on it in the theory chapter.

Laurenson’s 2005 article on the management of display equipment offers another methodology for conserving the equipment used to display time-based media artworks. Laurenson sets off from the significance (visibility, artist involvement, relationship to context and history) of the display equipment, and suggests a set of conservation strategies based on its reparability and availability (Laurenson 2005). This was relatively revolutionary when the article was published, as ethical guidelines such as the E.C.C.O Code of Ethics did not directly discuss the replacement of an artwork’s components (E.C.C.O. European Confederation of Conservator-Restorers Organisations 2003). The DOCAM decision tree (DOCAM 2010c) is likely based on Laurenson’s article, but extends display equipment to technological equipment in general. Based on the visibility and reparability of the equipment in question, it suggests the strategies of storage, migration, and emulation for equipment, not directly for software. What DOCAM means by emulation is not clear, as its definition quotes both the Variable Media Network’s definition (see section 2.4 in this dissertation) and the term used in digital preservation.[12] The decision criteria for how to handle the software following the migration or emulation of the computer hardware are not mapped out. The DOCAM approach begins with the hardware perspective. It selects

[12] “To emulate a work is to devise a way of imitating the original look of the piece by completely different means. The term emulation can be applied generally to any refabrication or substitution of an artwork’s components, but it also has a specific meaning in the context of digital media” (DOCAM 2010a).

a few significant properties as decision criteria for the replacement of hardware, but does not provide an evaluation of all of the artwork's defining properties. Laurenson developed the concept of work-defining properties for time-based media artworks only a few years later (Laurenson 2014, 92–94), inspired by the term significant properties used to describe digital objects in the field of digital preservation.

In the digital preservation field, the significant properties of digital objects are not the only criteria that have a bearing on digital preservation strategies. In addition to significant properties, Strodl et al. suggest using the characteristics of the preservation process, such as its usability, complexity, and scalability (Strodl et al. 2007, 33). These criteria apply to the preservation of software-based art too, although they are quite hard to fully satisfy given the diversity of software-based art. If conservation measures have to be repeated, however, their usability and scalability can be particularly relevant. Complex conservation interventions increase the risk of their being unsuccessful or more expensive than planned. Indeed, the cost of preservation actions is a major decision criterion that Strodl et al. mention. Its influence on the choice of preservation strategies cannot be underestimated. It is notable that whilst such costs are often mentioned in the digital preservation literature, they are almost never accounted for in conservation theory (with the exception of INCCA's revised decision-making model and Appelbaum's conservation treatment methodology). The question of costs even seems to be somewhat taboo in art conservation circles, with the E.C.C.O Code of Ethics suggesting that "the conservator-restorer must work to the highest standards regardless of any opinion of the market value of the cultural heritage" (E.C.C.O. European Confederation of Conservator-Restorers Organisations 2003).

General ethical principles that have been developed for the conservation of cultural heritage as a whole, such as minimal intervention and reversibility (E.C.C.O. European Confederation of Conservator-Restorers Organisations 2003), are also important decision criteria for the treatment of software-based art. It must be said, however, that for software-based art, reversibility is often not possible, as the hardware or the standards that undergird the software will have changed in the meantime. If the measures undertaken in conservation were reversed, the artwork would cease to function. As Katrin Janis writes in her doctoral thesis on conservation ethics, "For each intervention, the subsequent effects must be considered. Furthermore, it must be ensured that future restorations and investigations, possibly involving new questions, are not hindered by current interventions" (Janis 2005, 140). The latter requirement is just as difficult to guarantee as reversibility, as different

conservation strategies may involve interventions that hinder each other. Furthermore, it requires conservators to think beyond measures taken in the immediate present.

The influence of a particular institution's conservation policy on decision processes is not to be neglected either. Unsurprisingly, digital preservation policies are common in the digital preservation field. Due to the amount of content to be preserved and the diversity of formats, the emergence of policies that guide how material is supplied, preserved, and accessed is inevitable. It then follows that the automation of preservation processes is based on preservation policies (see for instance Becker et al. 2009, 5/25] or Simpson et al. 2019]). The purposing of a given collection as, for example, Evidence for an archaeological museum, or Display for an art museum (Keene 2002, 17), is an example of an institutional policy that impacts how artworks are preserved. In an archive, where objects are not being put to (their primary) use, they are evidence of the past. By contrast, the goal of contemporary art museums is to display functioning artworks and allow visitors a certain experience of them.

Moreover, conservation policy is closely tied to both conservation infrastructure and knowledge. Conservation infrastructure can also be a factor in decision-making, for instance regarding the type of virtual machine or the emulator to be used (Falcao, Ashe, and B. Jones 2014, 6).

This section has shown that there are many criteria other than an artwork's authenticity to be considered when making decisions about preservation measures and strategies. These criteria range from financial, legal, infrastructural, and technical considerations to institutional preservation policy. Most importantly, a long-term vision for durable, financially feasible, and reproducible conservation measures is lacking as a decision criterion in most conservation theory. This question of sustainability is especially crucial for software-based art, as its cycles of decay and preservation are shorter than for artworks made of most other materials.

2.6 Stakeholders

This thesis examines preservation strategies for software-based art, and the related processes of decision-making in museums and institutions. It is, therefore, worth briefly defining the term institution here. Institution has a broader meaning than contemporary art museum. By institutions, I mean those organisations that hold and care for collections of software-based art but that do not necessarily exhibit their collections regularly, or those that

do not describe themselves as museums. For instance, the House of Electronic Arts describes itself as a "platform for contemporary art that explores and employs new technologies" (House of Electronic Arts Basel n.d.a); it holds a collection of software-based art. LI-MA also has its own media art collection and conserves artworks for other collections as a service (LI-MA n.d.b), but has held only small-scale exhibitions over the last ten years. Rhizome "champions born-digital art and culture through commissions, exhibitions, scholarship and digital preservation" (Rhizome n.d.). For its exhibitions it collaborates with New York's New Museum.

The terms used to refer to those who experience and interact with software-based artwork also deserve attention. Such people can be visitors, users, viewers, the audience, or the public, depending on context: visitor and viewer tend to be used for visual art, audience for musical performances, user for software users (whether in an informal or business environment), and the public for different kinds of performance. As software-based art sits between visual art, performance, and games, any of these terms can be suitable.

I would also like to address the terms curator and conservator. These terms have different meanings in different languages, and it is therefore important that I define them for my purposes here. A conservator cares for and preserves artworks. A curator selects artworks for acquisition, makes suggestions for their display, and curates exhibitions. In the case of software-based art these roles may overlap. This is particularly frequent when determining a software-based artwork's significant properties, as the input of both conservator and curator is important.

The "network of care", a term coined by Annet Dekker, "is based on a transdisciplinary attitude and a combination of professionals and non-experts who manage or work on a shared project" (Dekker 2014, 81). Dekker highlights the fact that such networks usually form around artworks that do not belong to an institution. Nevertheless, I will use the term to refer to the team(s) responsible for preserving artworks for an institution. On one hand, this term makes clear that it is not only the conservator who conserves the artwork; on the other, it includes people outside of the institution. A network of care, therefore, includes not only conservators, and curators, but also technical specialists, as well as people who are not formally within the institution such as artists, and users engaging with the artwork. Networks of care are very common in the conservation of time-based media. In most museums, time-based media conservators and technical experts are external and called in on a case-by-case basis. However, these conservators often collaborate with technical experts, as the variety of technology that they encounter is broad,

and because most of them are not computer scientists or engineers (Jennings Lewis and Weidner 2013). Collaborations with users are rare in conservation, although many conservators (and artists) reuse code, software, and emulators written by people outside of their direct network of care.

Networks of care can be important not only for the conservation of the artwork, but also for the conservation infrastructure more broadly. An example of this is the Emulation-as-a-Service Infrastructure, which is used and maintained by a network of universities to make their digital assets accessible (Acker 2020] and section 2.4).

2.7 Knowledge Gap and Problem Definition

Having now discussed the state of the conservation and digital preservation field in relation to software-based art, and given the key definitions used in this thesis, I will summarise the knowledge gaps that I aim to address.

Software-based art does not often have clear boundaries, is frequently context-dependent, has many variable components, tends to multiply, can decay quickly, and is of liminal materiality. The fact that its significant properties can depend on its context and that they can be renegotiated for each preservation cycle makes identifying long-term solutions difficult and increases the effort required for documentation. This therefore raises the question of how a conservator is to deal with variable significant properties, and what constitutes sustainability in software-based art conservation. Should decision criteria other than the authenticity of an artwork be considered? As we have seen, other potential criteria include financial, legal, infrastructural, and technical considerations, and the preservation policy of an institution.

In order to fill the gap, mentioned above, left by the lack of long-term vision in the field of contemporary art conservation theory, this thesis will develop a theoretical framework for the conservation of software-based art that foregrounds such long-term perspectives. As such, it will aim to contribute new knowledge to the field of software-based art conservation.

Although the influential Variable Media Network operated at the crossroads between art conservation and digital preservation, it focused more on equipment than software. I will, therefore, investigate how the conservation strategies of the Variable Media Network might be expanded and made compatible with those developed in the field of digital preservation. For instance, the Variable Media Network definition of emulation collides with the one used

in digital preservation. A comparison of the strategies of reconstruction, as described on the DOCAM website (DOCAM 2010b), and documentation, as discussed by both Anne Laforet and Annet Dekker in relation to the conservation of net art (Laforet 2009, 186; Dekker 2014, 180), demonstrates, however, that the Variable Media Network's strategies are not complete.

My research will also look at how combining conservation strategies might benefit long-term preservation. I will focus on those strategies that can be executed at the software level; strategies for prolonging the life expectancy of hardware are not the subject of this research.

Furthermore, this dissertation will also analyse the strategy of emulation in more detail. Emulation has a large theoretical potential, as it replaces hardware that ages quickly. It encompasses a wide range of sub-strategies with different degrees of hardware dependency, each of which has its advantages and disadvantages.

Finally, a method is required for identifying which conservation strategies are technically possible for a specific software-based artwork. As such, I aim to build a framework that facilitates the identification of conservation strategies for software-based art and when to apply them.

My overall aim, therefore, is to propose a theoretical framework for the conservation of software-based art that facilitates the identification of sustainable conservation strategies, that integrates institutional, economic, and socio-technical considerations, and that accounts for long-term perspectives within conservation.

2.8 Methodology

Software-based art lies between contemporary art conservation and digital preservation. Conservation practices and methods from both fields are therefore of interest to us here. I am applying what is, for art conservators, the familiar case study method. This approach allows me to examine conservation strategies in depth, and to support conservators in identifying strategies that are suitable for long-term preservation. As I am interested in the conservation of software-based art in an institutional context, I have chosen two software-based artworks that belong to institutions, and that represent different types of software-based artworks. Because institutions are rarely happy to let researchers do hands-on research on their artworks by applying conservation strategies to them, the question of availability limited my choice of case studies significantly.

As I conducted my research residency at LI-MA, an institution in Amsterdam that researches, distributes, and documents media art, I became involved in the project *Future Proof* (LI-MA 2016-2017b) whose goal was to make the software-based artworks of Dutch artist Geert Mul more sustainable. Hence, I chose Mul's *Horizons* (2008), owned by the Museum Boijmans Van Beuningen. The Museum Boijmans Van Beuningen is an art museum in Rotterdam with a collection that dates from the Middle Ages to the present day. *Horizons* allows the visitor to experience the museum's landscape painting collection in an interactive way. When a visitor enters the space, a loud noise is immediately emitted that reacts to their movements around the space. An immersive video projection that covers an entire wall also reacts to their movements. This is made possible by a sensor that tracks the visitor, and influences the projections of the digitised paintings and the sound in response to this sensor input. The occasion for me to study *Horizons* came about in the context of a Geert Mul retrospective exhibition, planned for 2017. Mul originally intended to exhibit *Horizons*, but the Museum Boijmans Van Beuningen did not have the budget to purchase the necessary hardware such as the computer and tracking sensor.[13] Instead, Mul decided to exhibit *Shan Shui* (2013), which belonged to him. *Shan Shui* is based on the same software as *Horizons*, but differs in the image database used as well as the hardware setup. Whilst *Shan Shui* and *Horizons* both use the same tracking sensor, the same computer architecture, and the same projectors, *Horizons* uses three video projectors and *Shan Shui* only requires two. These input and output devices are all mass-produced and, as such, represent a broader category of stand-alone (that is, contained; see section 2.1) interactive software-based artworks.

For the second case study, I selected an internet-based artwork owned by the House of Electronic Arts, Basel. The House of Electronic Arts is small compared to the Museum Boijmans Van Beuningen, but specialises in born-digital art and owns more such artworks than many larger contemporary art museums. I was working for the House of Electronic Arts as a freelance conservator, and this enabled me to choose an artwork from their collection (House of Electronic Arts Basel n.d.b). The artwork *TraceNoizer* (2001-2004), a website made by the artist group LAN, was defunct and no longer online when the House of Electronic Arts acquired it in 2017. *TraceNoizer* is a dynamic website that lets users create (different) clones of other websites. Through the proliferation of these fake websites, the

[13] The artwork was a gift by Geert Mul and was never transferred to the museum (Mul, Zijlstra, and Kisters 2017).

originals can no longer be found by the search engine, and thus become hidden in plain sight. As a dynamic website, *TraceNoizer* is not dependent on specific hardware, but is heavily dependent on its external networked environment.

I conducted technical analyses of both case studies, and used them to test and compare conservation strategies. For each artwork, I applied two different conservation strategies. To carry out these strategies on the first case study, I requested the support of Teal Gaure, a software engineer. For the second case study, I had support from Rafael Gieschke and Klaus Rechert, both computer scientists at the University of Freiburg, who executed the emulation strategy (network of emulators). Fabian Thommen, one of the artists in LAN, also provided help with the migration strategy. This hands-on approach, in contrast to an archival one, was important for me, as whilst I am trained as a time-based media conservator, I did not have much knowledge of computer science and software engineering. During my research, I refreshed my knowledge of programming by taking a course in the C language which helped me to understand the artworks' source code and how it was compiled. A minimal technical knowledge is crucial for being able to understand how specific conservation strategies may be identified, and for understanding the nuances between sub-strategies. I complemented my research on the case studies with an artist interview for *Horizons* and *Shan Shui*, a visit to Geert Mul's studio, an artist interview with LAN for *TraceNoizer*, two interviews with the programmer of *Horizons* and *Shan Shui*, and an interview with Geert Mul as well as two collection care professionals from the Museum Boijmans Van Beuningen. I also had the opportunity to observe the artist preparing *Shan Shui* for exhibition at the Stedelijk Museum Schiedam (NL).

In addition to the two case studies, I can also bring my professional experience as an assistant time-based media conservator at Tate from 2014 to 2016, and as a conservator for the House of Electronic Arts where I am responsible for collection care. My work experience has allowed me to encounter many other software-based artworks that I did not have the opportunity to study in depth, but that confirm that my case studies are representative of the kinds of issue that other conservators of software-based art face. I was also able to see the difference it makes when a professional time-based media team takes care of artworks, as opposed to other museums where there are no time-based media conservators. For example, some of my potential case studies turned out to be unsuitable, because the collection that held them had only kept the sculptural parts and not the computer with the software. This made it impossible to apply different conservation strategies to the software-based part of these works and compare it with the original.

I conducted the literature review on literature for both art conservation and the digital preservation of archives and libraries. For my theoretical framework, I consulted literature from beyond the field of conservation, such as science and technology studies, as well as literature about sustainability with regards to ecology. Based on my experiences with the case studies mentioned above, I infer a theory for the conservation of software-based art and an approach to identifying conservation strategies (Chapter 3). I analyse the long-term effects of different conservation strategies on the artworks that constitute my case studies (Chapters 4 and 5). I then examine whether the application of my theoretical framework results in useful practical recommendations for the conservation of software-based art (Chapter 6). From this, I draw conclusions about the application of my theoretical framework and point to opportunities for further research (Chapter 7).

3 Theoretical Framework: The Artwork as Technosystem

This chapter sets out the theoretical base by which the various preservation strategies were selected, applied, and evaluated in the case studies. Its goal is to analyse and conceptualise software-based artworks in a way that facilitates the selection and evaluation of sustainable preservation strategies.

The chapter begins with an investigation of the term sustainability as it is used in nature conservation and digital preservation, and develops criteria for the sustainable preservation of software-based artworks (section 3.1).

Section 3.2 discusses the timeframe for the preservation of software-based art in this research, and relates the longevity of particular instantiations of such artworks to changes in their wider technological environment. Due to the short lifecycle of much hardware, the concepts of instantiations and versions are crucial to understanding the long-term preservation of software-based art.

Section 3.3 shows how Feenberg's framework of the technosystem can be meaningfully applied to the preservation of software-based art. This framework enables those in charge of an artwork's care to identify sustainable preservation strategies, as it incorporates not only the technical environment, but also institutions, their resources, infrastructure, and the audience/users.

In addition to the technosystem, Dourish's concept of the materiality of the digital, as discussed in section 3.4, offers an account of how materiality is expressed in software. An attention to materiality is important when identifying the significant properties of a software-based artwork, because the material may itself be the carrier of meaning. Section 3.5 explains the significant properties through the lens of the technosystem.

Section 3.6 operationalises the theoretical framework developed in the previous sections by describing preservation strategies from the perspective of the technosystem.

Finally, section 3.7 suggests ways to map a technosystem onto a software-based artwork so as to model sustainable preservation strategies. This final section will support both the analysis and hands-on preservation of the case studies as described in chapters 4 and 5.

3.1 Sustainability Criteria for the Conservation of Software-Based Art

Contemporary artworks, including software-based art, are often not made with long-term sustainability in mind (see, for instance, Lozano-Hemmer 2019, 105]). The terms ephemeral,[14] performative,[15] and processual[16] are often used to describe contemporary art, and they hint at its fleeting character (see, for instance, Dominguez Rubio 2020, 85], and Engel et al. 2018, 17/41]). However, when such works are acquired by museums or other cultural heritage institutes, they enter a domain in which their long-term preservation is a core responsibility. Museums, therefore, need sustainable preservation strategies in order to look after their artworks in the long term.

As sustainability has not been explored in relation to conservation strategies, I will begin by examining other domains before determining its meaning in the realm that interests me here. Sustainability is currently a fashionable term used in many different fields, each with very different meanings. Definitions of the term as used in environmental studies and digital preservation are the most fruitful for my purpose, and it is those that I will draw on.

In environmental studies, sustainability has to do with the depletion of natural and economic resources. Sustainable development as defined by the United Nations presupposes that the replenishment of natural and financial resources matches the rate of their consumption, so that future generations are not disadvantaged (United Nations 2014, 8). The equal distribution of these resources, and of any environmental damage that ensues, amongst society is another requirement of sustainable development. To illustrate this, a three-pillar model is generally used: the natural environment, the economy, and the society must be in equilibrium (see Figure 7).

---

[14] (Caianiello 2013).

[15] According to the conservators Louise Lawson and Deborah Potter, contemporary art is shifting towards "kinetic, electrical, lighting, audio, performative and software based works" (Lawson and Potter 2017, 130).

[16] (Vall 2017, 84).

Figure 7: Logo for the Conference on Sustainable Development in Rio de Janeiro in 2012, symbolising the three pillars of environment (leaf), economy (stairs), and society (person).

The model of sustainable development assumes economic growth, as symbolised by the stairs in Figure 7. Although each new instantiation of a software-based artwork introduces changes to the work and increases its variability, it is not an objective of either the institution or the conservator to foster growth and diversity within the same artwork. On the contrary, they aim to keep such growth and new complexity to a minimum, and to stabilise or even freeze the work in time. As the complete freezing of a work is usually not possible or desirable (change might be part of the artwork), certain dynamics or dynamic equilibriums have to be accepted. Indeed, conservation sits between these two conflicting priorities of development/progress and preservation/freezing. Whilst conservation must adapt the artwork to contemporary contexts, it also strives to slow down the processes of ageing and obsolescence. Thus, conservation strategies are situated in a spectrum between, on the one hand, preservation without changes and, on the other, the development and continuous adaptation of an artwork to its changing environment. The assumption of constant growth, therefore, does not apply to the conservation of artworks, and nor do concerns about scarce material and energy resources. The reliance of software-based artworks on electricity and electronic equipment is already a given, and additional material and energy consumption for their conservation are not usually scarce. The long-term perspectives and idea of a balance between different realms of action are, however, directly transferable from the realm of ecological conservation to the concerns of the present study.

One notable instance of the adoption of the need to balance different realms of action was by the Digital Curation Sustainability Model, developed for libraries and archives in 2015 to provide support for the preservation of their digital assets. The model also follows a three-pillar system and assumes that sustainable preservation is only possible with a healthy institution, stakeholders who invest in it, and digital assets that have a value for both the institution and its stakeholders (Figure 8).

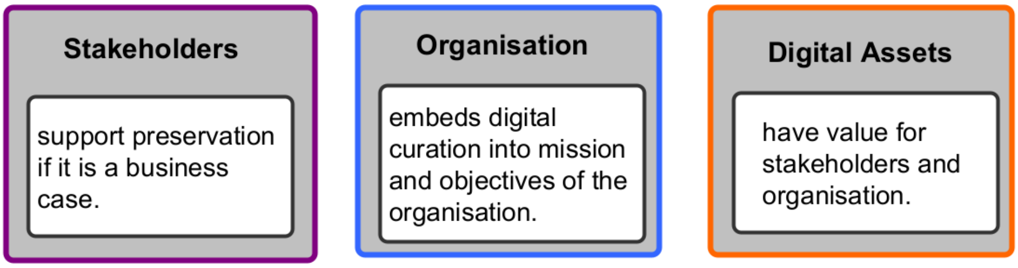


Figure 8: The Digital Curation Sustainability Model (2015)

The model assumes that financial resources are limited, and that this will directly impact the long-term preservation of an institution's assets. The last pillar indicates that preservation strategies can be considered sustainable if they are able to preserve the digital assets' value. This digital curation sustainability model translates the economic use of limited natural resources to limited financial resources, which indirectly includes human resources and infrastructure. It implies that institutions have to invest regularly in their assets in order to maintain or increase their value. "The Digital Curation Sustainability Model proposes that the value of digital assets can only be realised over time if resources are found to support their curation and this highlights a problem that all curation services have" (Grindley 2015, 21). This requires regular, long-lasting financial investments, which are frequently denied by governmental and institutional funding bodies that issue funds based on projects.

The Digital Curation Sustainability Model is primarily intended for the managers of libraries and archives. However, as my research addresses the conservators of artworks in museums, this model requires adaptation in order to reflect the specific tasks of this group of professionals. Therefore, for the conservation of software-based art, I propose the following three pillars: the network of care, the institution, and the conservation strategies. Each of these realms of action needs to be sustainable or act sustainably to ensure the long-term conservation of software-based art. I consider conservation strategies to be sustainable if:

a) They guarantee access to the artwork and its documentation and ensure the readability of the artwork.
b) They lower, or at least do not increase, the efforts required to preserve an artwork for the next preservation cycle, for instance by reducing its structural complexity, or by stabilising its source code complexity.
c) They are scalable.
d) They allow future generations to preserve the artwork based on new technologies and knowledge.

e) The network of care stabilises the significant properties of the artwork to a certain extent.

f) The network of care documents the preservation measures and related decisions transparently.

g) The institution is able to provide a conservation budget and policies that enable the perpetuation of the artwork in the long term.

These non-hierarchical criteria must be fulfilled cumulatively. Criteria a) to d) apply to all conservation strategies, with the exception of reinterpretation and continuation. As indicated in section 2.7's definition of the problem, I do not discuss these latter strategies here because they generate new software. As such, they would be better suited to a set of sustainability criteria designed for software design (see, for instance, Robillard 2016]). The following figure summarises the sustainability criteria.

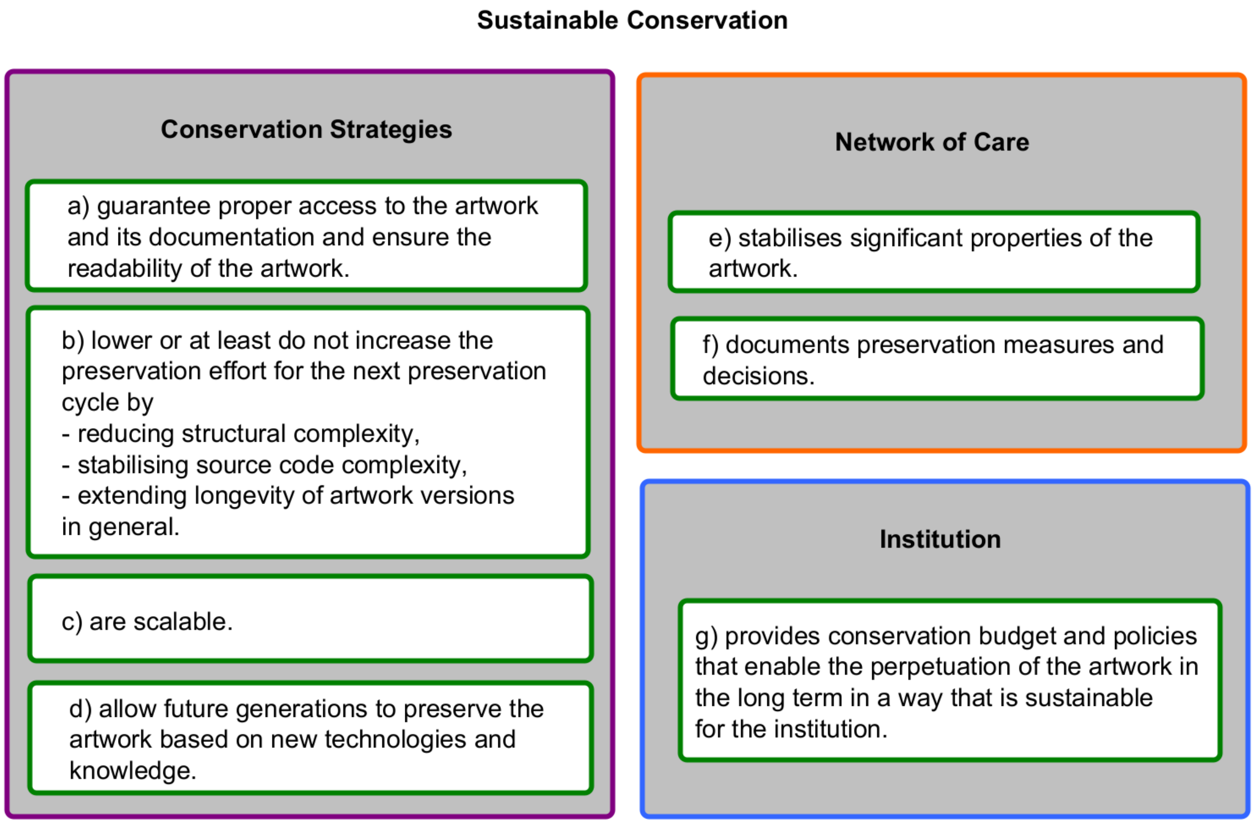


Figure 9: Sustainability criteria for conservation.

I will now briefly substantiate each sustainability criterion, and illustrate how it affects the conservation of software-based artworks.

a) Conservation strategies are sustainable if they ensure access to and readability of the artwork and its documentation.

The requirement that the public must be able to access documentation is based on the Venice Charter (ICOMOS 2nd International Congress 1964), a document that established conservation ethics for the museum community. Making art accessible is a stated mission of every museum. Ensuring access to and readability of artworks decreases the likelihood that they disappear into storage to be forgotten. A simple installation process, easy digital access, and documentation that helps viewers to understand the artwork all contribute to this aim. Whilst permanent access to artworks might be difficult to realise due to their fragility or the installation effort, making documentation publicly accessible is far more feasible. Lowering the barriers to documentation improves communication between the collecting institutions and researchers, the audience and external conservators, and amongst collecting institutions. It also fosters informed preservation decisions and discussions. At present, it is highly unusual for a museum to share conservation documentation publicly, although certain museums allow visitors to view documentation onsite upon request.

b) Conservation strategies are sustainable if they lower, or at least do not increase, the efforts required to preserve the artwork for the next preservation cycle.

Many institutions struggle with the maintenance and conservation of media art. This is caused not least by the difficulty of gaining funding for these activities.[17] Economic preservation strategies mean that they can more easily be repeated in the future, and that the organisation can provide resources for regular maintenance. Single, very expensive measures can only be justified economically if they reduce the financial burden of the following conservation cycle(s) considerably. The reduction of structural complexity, by for example freezing and replacing external dependencies, would constitute such a measure. As a consequence, future maintenance is reduced, and the artwork has a better chance of surviving for longer. Naturally, the number of dependencies can only be reduced if the significant properties of the artwork allow it, or if there is no alternative because the external dependency is already broken and cannot be reinstated. Another important point to consider is that complex maintenance increases the risk of introducing errors. The impossibility of creating a complete test that checks both the functionality and aesthetics of an artwork (Kraner 1997) is a major argument for reducing the complexity of maintenance. Another method for decreasing structural complexity is the generalisation of technical components

[17] See, for instance, (Evens and Hauttekeete 2011, 163): “Owing to the shortage of funding, archival materials suffer from deterioration and technological obsolescence. In this context, cultural heritage institutions are confronted with several major challenges that hamper the digitization and preservation of cultural heritage.”

such as computers, which also facilitates the sharing of software environments and emulators. This makes the artwork less hardware-dependent. Furthermore, preservation strategies should not increase source code complexity. Belady and Lehman found that “as a large program is continuously changed, its complexity, which reflects deteriorating structure, increases unless work is done to maintain or reduce it” (Belady and M.M. Lehman 1985, 253). It is easy to increase source code complexity involuntarily through conservation measures such as updating, migrations, and adaptions. With the growth of source code complexity, both the risk of conservation errors and the conservation effort are liable to increase. Finally, preservation costs can also be reduced through low resource consumption, such as taking up little digital and physical storage space, and low energy consumption, for instance by using offline storage such as digital tapes.

c) Conservation strategies are sustainable if they are scalable.

Making conservation strategies scalable allows for the reuse of research that was carried out into them, for the sharing of their implementations, and maintenance; this greatly increases the chances of sustaining software-based artworks for a longer period. Instead of treating an artwork individually, an artwork is for instance wrapped in an emulation that can then be managed and maintained along with other emulations. Whilst the application of the scalability criterion depends on other artworks, it forces the network of care to consider other artworks in the collection that could be treated together.

d) Conservation strategies are sustainable if they allow future generations to preserve the artwork based on new technologies and knowledge.

To anticipate and include the needs of future generations in today’s decision-making is a requirement of the United Nation’s definition of sustainability (United Nations 2014, 8). A similar requirement comes from the field of conservation ethics. As Katrin Janis states: “For each restoration, the subsequent effects must be considered. Furthermore, it must be ensured that future restorations and investigations, possibly involving new questions, are not hindered by current interventions” (Janis 2005, 140). This challenging requirement has implications for the acquisition process. The institution must make sure that the material necessary for the preservation of an artwork such as source code or disk image is acquired, and a thorough description is made of the artwork, its meaning, software, and hardware. This enables future generations to understand the artwork, and allows them to apply their own preservation measures. Moreover, the acquisition budget should not only include the acquisition cost of the artwork, but also consider conservation, maintenance, and documentation costs.

e) Conservation strategies are sustainable if the network of care stabilises the significant properties of the artwork to a certain extent.

The aforementioned sustainability criteria do not solve the problem of subjective differences regarding an artwork's significant properties (see section 2.3), and hence the risk that these be renegotiated with every conservation cycle. Such a scenario would potentially lead to frequent changes of conservation strategy, thereby introducing new costs and risks with every cycle. This makes the requirement to stabilise (to a certain extent) a work's significant properties necessary, so as to enable sustainable preservation measures. Although it makes sense to check and adapt such significant properties recursively, as Castriota suggests (2019b, 171), this should not make it necessary to completely change the preservation strategies with every preservation cycle.

f) Conservation strategies are sustainable if the network of care documents preservation measures and related decisions transparently.

This requirement for sustainable conservation strategies originates from the field of conservation ethics: conservation measures and decisions must be documented in a way that allows future conservators to understand why they were taken (Cologne Institute of Conservation Sciences / TH Köln 2019, revised 2021, 14; Appelbaum 2007). For instance, if interventions are done at the level of the source code, they should be documented, in the form of comments within the source code, or as a description of commits of changes within a versioning control system (Engel and Phillips 2019, 188). This supports maintenance work and future preservation strategies, and clarifies which changes have been made by who, and why.

g) Conservation strategies are sustainable if the institution is able to provide a conservation budget and policies that enable the perpetuation of the artwork in the long term.

This requirement derives directly from the Digital Curation Sustainability Model mentioned earlier (Grindley 2015, 21). If conservators do not have an appropriate budget and/or infrastructure for the preservation of software-based art, they cannot preserve them in the long term.

These criteria for the sustainability of preservation strategies will be applied to the case studies described in the following chapters, and will allow me to compare different strategies as applied to one artwork. The following section looks at the life cycles of software-based artworks, as the relatively short longevity of an artwork's instantiation is

unique to software-based art and represents a significant challenge for long-term preservation.

3.2 Instantiations, Versions, and Life Cycles of a Software-Based Artwork

Time is a very important factor for software-based art, not only for its functioning and processing, but also its preservation. It is difficult to imagine how long such artworks might be preserved, given the rapid development of both technology and society. This section will discuss the life cycle of a software-based artwork and define the terms instantiation and version.

The longevity of a software-based artwork can be understood as the average lifespan of its versions. Before the first stable version is acquired by the museum, an artist might have made changes to the work's hardware and/or software, or further developed it. The museum might acquire one of these changed versions (see Figure 10). It will have documented it and, after a certain time, made adaptations and replacements in order to preserve it, resulting in a second version. Quite quickly, therefore, the artwork could be presented using equipment different to that originally acquired, and possibly even embodied in several instantiations or versions simultaneously. Here I follow Castriota's definitions of the terms version and instantiation: an instantiation refers to a discrete occurrence of an artwork version (Castriota 2019b, 74). A version, meanwhile, refers to "various subtypes, which may or may not be tokened in time and space through a concrete manifestation" (Castriota 2019b, 75).

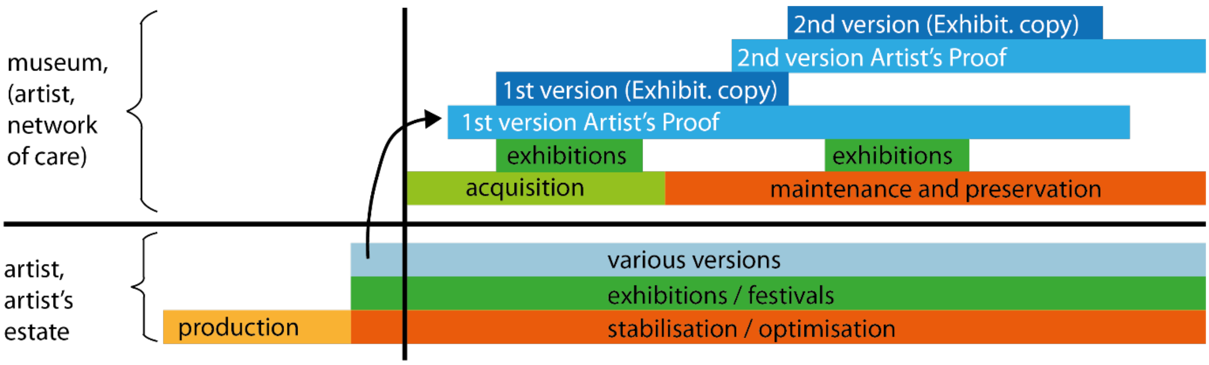


Figure 10: Possible instantiations of software-based artworks before and after entering a museum. The instantiations are called versions, as they are updated/adapted in order to keep the work functional.

After a period of time, the artwork may fall into neglect, become damaged, or prove too difficult to maintain technically due to external dependencies. Hence, the museum or the

network of care might decide that the work's documentation becomes itself the artwork, or that its dysfunctional remains might represent it. Figure 11 represents the life cycle of an artwork and includes the following steps: creation, optimisation/stabilisation, perpetuation (through preservation strategies), ageing and/or negligence, becoming dysfunctional, and finally de-accession or being forgotten. Around the perpetuation stage, the artwork may enter into a spiral of different versions.

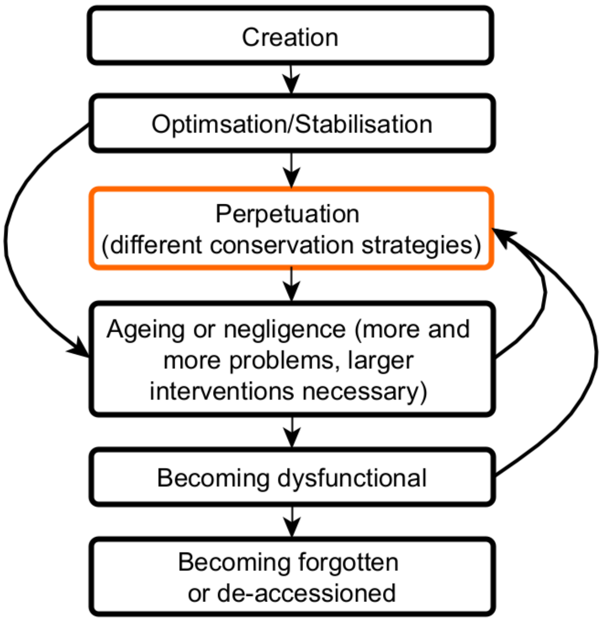


Figure 11: Life cycle of a software-based artwork.

The longevity of a version of a software-based artwork is generally between one and ten years.[18] The longevity of internet-based artworks with external dependencies is often shorter, whilst that of contained artworks depends on how often they are on display, as computers with long operation hours age more quickly. However, how long an artwork can be preserved is not only a question of the longevity of its versions. Changes in technology also have a huge impact, as they can make creating a new version more difficult. For example, if the hardware is difficult to replace because only a handful of spare devices are left, such as in the case of *Super Mario Clouds* by Cory Arcangel, which requires an obsolete Nintendo console for playback (Magnin 2016, 13), conservators will try to extend

[18] This is based on my own experience. Internet-based artworks, in particular, are prone to change. The replacement of a web service or the updating of an API can cause a work to lose functionality after a little as a year, sometimes even sooner. So-called contained artworks, which do not rely on internet access, have longer life spans, and depend mainly on their hardware working properly. Depending on how frequently the hardware is used, it can last as long as two decades for artworks that are never on display. Hence ten years as a probable value for an artwork that is not frequently exhibited.

the life of the hardware. If the hardware cannot be saved, it will either be reconstructed or emulated; failing this, the artwork will be documented or reinterpreted using a different technology, or de-accessioned, as it is at the end of its lifecycle.

It is useful to think about the rapidity of technological change to have an indication of how long one might be able to perpetuate a software-based artwork. The time span of a media technology such as analogue film is about one hundred years, analogue video fifty years, Compact Disc technology thirty years, computer punch card technology thirty years, internet as a network technology thirty years and on, and personal computers thirty years. The rate of change for new digital formats or software is about two to ten years, for new operating systems three years, for new digital hardware one year, and the lifespan of digital hardware is generally three to ten years. Moreover, these transformations are accelerating, and the duration of new technology is becoming shorter. Over a period of twenty-five years – one generation in human terms – a software-based artwork will go through several instantiations. However, conservators do not usually conceptualise preservation in terms of number of instantiations; according to Hölling, "in conservation and museum practice, the life of a conservator or a curator is too short to grasp the temporal passing of a masterpiece, which is therefore conceived – and has to be conserved – to endure forever, or at least for an 'ever' of a human temporal dimension" (H. Hölling 2016). Hölling argues that this "elicits the idea of a stable, 'conservable' object and what determines traditional theories of conservation." An awareness that an artwork will have to go through many different instantiations and versions, and requires the active involvement of the conservator (as opposed to providing simply adequate storage conditions), might give rise to different preservation strategies. Thinking long-term and acting short-term is crucial in the conservation of software-based artwork.

Now that the criteria for sustainable preservation strategies have been established and its temporal horizon discussed, the next section explores the conceptualisation of software-based artworks as technosystems, in order to provide a theoretical framework for decisions about sustainable preservation strategies.

3.3 The Technosystem as a Model for the Conservation of Software-Based Art

Because software-based artworks perform in an ever-changing environment, they are susceptible to breaking down. The framework that I require has to be able to illuminate the

specificities of preserving software-based artworks, a process characterised by a multitude of technical, social, and human actors, all with a variety of interdependencies. Such a framework must explain why artworks break down, and lead to sensible and sustainable preservation decisions. It therefore needs to be applicable to the whole range of software-based artworks, for instance, from contained to networked (see section 2.1), from single author to collaborative, from non-interactive to interactive. It must also integrate human actions such as artists, the public, conservators, and other stakeholders. As these stakeholders handle the artwork in a variety of technological, social, institutional, and market-oriented environments, this too should be represented in the model. Software-based artworks have a material side: they depend on their equipment as well as other infrastructures and technical environments, such as the internet. Andrew Feenberg's concept of the technosystem aptly captures the material, technological, and social dimensions of software-based artworks, and is capable of representing conservation in an institutional environment with limited resources.

Feenberg is a philosopher of technology and investigates the overlap between critical theory and science and technology studies. In 1991 he developed the *Critical Theory of Technology* (Feenberg 1991; Feenberg 2005) in which he combined the concept of technology as a neutral, technical discipline exercised by engineers, technological determinism, and technology as an embodiment of social processes (social constructivism) in his "instrumentalisation theory." In his book *Technosystem* (2017), he goes one step further: as a social constructivist and philosopher of political theory, he suggests using technology to democratise and thereby improve society and our living environment more broadly.

Feenberg uses the term technosystem to describe "a field of technical practices aimed at control of the environment, whether natural, economic, or administrative. To that end the environment is interpreted and structured as an ensemble of sociotechnically rational functions" (Feenberg 2017, 159). He introduces the concept to further develop his instrumentalisation theory, in order "to open up the imagination to a possible transformation of industrial society" (Feenberg 2017, 186). The technosystem is based on the idea of the "coproduction of society and technology in the making of worlds". In other words, "The design process brings together meaning and matter. It is a terrain on which social groups express their worldview materially and advance their perspectives and their interests." (Feenberg 2017, 32).

My research does not ask the same questions as Feenberg, such as how to use technology to improve and democratise society. However, his notion of society and technology as ineluctably intertwined such that together they make the world is relevant to conservation, as the objects of conservation are not merely material or technological matter, but also representations of ideas and expressions of social processes. The notion of progress is important nonetheless, as it is this that causes friction between the object and its (ever-progressing) environment.

As my research focuses on the conservation of objects, I primarily draw on Feenberg's discussion of how an object's function emerges as a result of technical and social design processes. These processes are both influenced by the context in which the artist lives and works in, which is then expressed through the artwork. Each intervention to preserve, renew, migrate, or actualise the artwork will move it further away from this context, but also enable the artwork to continue functioning, therefore keeping it as close as possible to the artist's original intention.

Feenberg uses the term layers of function to describe the technosystem. These layers prove very useful for discussing information technology and artworks: "Design proceeds through bringing together layers of function corresponding to the various meanings actors attribute to the artifact. The study of the technosystem must identify the layers and explain their relations" (Feenberg 2017, 32). Layers are an important characteristic of a technosystem, and enable us to understand and describe it. A typical example is the internet, with its layers of hardware infrastructure, software infrastructure, service and platform layer, and finally its use layer in the hands of social groups and business. These layers are often co-dependent on one another, and being aware of them not only provides a better understanding of the system but opens new approaches for preservation strategies. Section 3.7 describes how layers are manifest in software-based artworks.

Feenberg sees a subject not as isolated, but as situated in and related to an environmental niche. The tension between a thing and its environment is typical of a technosystem and the driving force behind its evolution (Feenberg 2017, 75). However, an artwork's changing environment is also a common reason for it to become dysfunctional due to the manifold dependencies between it and its environment. Software-based artworks are usually complex, dynamic, and open systems. For instance, in internet art created before 2000, hyperlinks leading to other websites were a typical and important feature as search engines were not as efficient as they are today (Lialina 2009, 27). However, over twenty years later, these external websites are either offline (link rot), web browsers cannot open

them because they are not encrypted, or their content has changed completely (reference rot). The conservator therefore needs to draw the boundaries of the website-system wide enough and devise a strategy for dealing with these linked websites. The concept of the technosystem can serve to map the changing net environment of an artwork and locate its environmental niche.

Not only might the environmental niche that is preserved with the artwork change, but so too might its institutional environment. When an institution acquires a web-based artwork, it must consider the maintenance of both the web server and the artwork. Who maintained it before and how can this be continued? A web server's maintenance team, the digital infrastructure, and the conservation budget are all factors that have an impact on the preservation of an artwork (besides others). Feenberg explains the position of institutions and artefacts in relation to the technosystem:

> Institutions and artifacts consist in assemblages of components joined by their functional role in society rather than by an intrinsic essence. The *technosystem* resembles a palimpsest: multiple layers of influence coming from different regions of society and responding to different, even opposed, logics inscribe a shared object or institution. (Feenberg 2017, 26)

Institutions are examples of administrations which – along with markets and technologies – are components of the technosystem (Feenberg 2017, x). They classify objects and treat them uniformly following certain rules (Feenberg 2017, 22). Integrating the institution within a theoretical model makes sense in the context of sustainable preservation strategies where financial resources and know-how play an important role. Feenberg's theory of the technosystem also allows the role of the market to be introduced as a factor in conservation. Indeed, conservation decisions are influenced by the conservation budget of an institution, which is also taken into account in my definition of sustainability (see section 3.1.) which, as we saw, holds that conservation measures should lower the effort required for the next preservation cycle.

Feenberg does not refer to systems theory in *Technosystem*. As technosystem is a very comprehensive term, one might imagine that he does not want to define systems too rigidly. Although he uses the term system frequently in his book, this is not to define the technosystem. For Feenberg, the technosystem is "the field of technically rational disciplines and operations associated with markets, administrations, and technologies" (Feenberg 2017, x). In this definition he uses the terms field, operations, and disciplines to

describe the technosystem. In the definition of artefacts and institutions quoted above, he uses terms such as assemblage, layer, palimpsest, and components. In contrast to these terms, systems theory uses terms from the natural and information sciences such as feedback loop, complex, open/closed, dynamic/static, emergence, boundaries, environment, and input/output. Feenberg's terms appear to stem from the humanities, and are less technical than those of systems theory. Here he is in good company, with literary scholar Katherine Hayles and anthropologist Anna Lowenhaupt Tsing using assemblage instead of system to indicate that the relations between components are either too complex to grasp or hazy and unknown (Brain 2018, 7). As my research takes place in an interdisciplinary field that straddles conservation, computer science, and the humanities, I will use terms from all of these disciplines. A conservator's description of an artwork's system will depend on the condition that it is in. If it is heavily damaged and in something of a state of ruin, it might be described as an *assemblage* of fragments. If it is in good condition, still able to function, and its behaviour cannot be predicted from the sum of its single elements, it might be described as a *complex system*. Of course, an artwork's complexity and the ease with which it can be understood are relative. They depend on the knowledge of the conservator as well as the knowledge base of the institution, or the relevant "network of care."[19] Describing an artwork as a system enables or enhances our understanding of it along with its components, dependencies, and functionality, whilst also acknowledging the system's simplifications.

This section has described the technosystem as a framework for the conservation of software-based art with the following characteristics:

- Artefacts are the result of a process of technical and social design which brings together meaning and matter. It is, therefore, important to understand an artwork's materiality.
- Institutions enable but also limit the conservation of their objects.
- Art markets, technology markets, and economic considerations play a considerable role in conservation decisions. This is reflected in the sustainability criteria explored in section 3.1.
- Artefacts consist of layers of function and of layers of influence. Recognising these layers can help conservators to identify preservation strategies.

[19] "A 'network of care' is based on a transdisciplinary attitude and a combination of professionals and non-experts who manage or work on a shared project" (Dekker 2014, 81).

- Artefacts are located in a particular environmental niche. Often, this niche is as important for the preservation process as the artwork itself.

3.4 Dourish's Concept of Materiality Applied to Software-Based Art

As early as 1996, the media theorist and art critic Boris Groys recognised that restoration (a form of conservation) is an interpretative process. He states that a conservator must decide – or interpret – whether the restitution of an artwork's identity is based on material continuity or form (Groys 1996, 156). In the case of software-based art, this dilemma immediately comes to the fore. In addition to the rapid obsolescence of software, one of the main causes for the breakdown of software-based artworks is the ageing of their computing devices and other electronic equipment. Although Feenberg refers to materiality and production processes as part of the technosystem, as a philosopher, his contribution does not include any actual research into materiality per se. In order to address this gap and properly investigate the question of artworks' materiality, we may turn to Paul Dourish's book *The Stuff of Bits* (2017). Dourish, a computer scientist, uses anthropological and ethnographical methods for his research, which looks at areas such as human-computer interaction (HCI), computer-supported cooperative work (CSCW), and science and technology studies (STS) (Dourish n.d.). *The Stuff of Bits* confirms the importance of materiality not only for hardware, but for software too. Dourish's description of information technology sets off from its material side, and examines the effects of this materiality on the ways in which we perceive and use certain information: "I take it as a premise that social and material are entwined, and that the materiality of the digital shapes the cultural experience we can have of it and the purposes to which we put it" (Dourish 2017, 26/190). As such, Dourish analyses the materiality of digital artefacts in order to "reassess the relationship between the technical and the social" (Dourish 2017, 38/190). Because software does not have a purely functional value, but also mirrors social relationships and processes, this analysis is relevant to the process of identifying an artwork's significant properties. In particular, as time passes, software can acquire historical and socio-cultural values through associations with certain uses, be they industrial or economic, or with certain communities. Dourish's contribution is, therefore, of great significance for any theory of software-based art conservation.

For Dourish, programming can be compared to a handicraft. Programmers may have a relationship to the software that they shape much like a sculptor may have to bronze: “Programmers understand the ways in which digital structures can resist their will, every bit as much as clay, wood, or stone” (Dourish 2017, 12/190).

Like Feenberg, Dourish highlights the different layers of digital infrastructure that come to shape our experience: “the materialities of digital information often lie hidden multiple levels from view” (Dourish 2017, 27/190). Indeed, although these layers are often not visible, they are important in shaping our perception of the software-based artwork. Engel and Philips support this argument: “Similar to other areas of art technology, these choices of medium can be deliberate (artist-intended), or contingent. In both cases, the source code may be integral to an artwork’s identity, even if it is typically hidden from the audience and rarely part of the audiovisual or interactive experience” (Engel and Phillips 2019, 181).

Lance Strate, a scholar of media studies and media ecology, highlights the role played by media in perception: “We mediate with our bodies, our sense organs and nervous systems. We mediate through our languages, art forms, and symbol systems. And we mediate through our technologies, techniques, and technical systems” (Strate 2010, 37). The choice of a particular piece of technology for an artwork is likely a highly conscious one designed to best transmit the artist’s intention for the work. The technology chosen will have certain properties that enable the artist to articulate meaning; as such, it is through these properties that a viewer/user will perceive the artist’s ideas. This is an important aspect of conservation, as these features must be identified.

Dourish also recognises the variability of digital artefacts that comes as a result of their variable physical and digital environment: “The materialities of digital systems lie not least, then, in this gap between what is denoted and what is expressed, or between the specification and the execution. A computer program may be a precise series of instructions, and yet the experience that results from the program’s execution is radically underspecified by that program” (Dourish 2017, 24/190). In 2006, Pip Laurenson discussed the specification of time-based media artworks and the resulting variability with reference to the philosopher Stephen Davies:

> Davies uses the terms ‘thin’ and ‘thick’ to indicate the degree to which the composer has determined the detail of the performance through work-determinative instructions, for example, how much direction the composer has given to the

> performers via the score and the degree of additional direction regarding instrumentation. The concept of a ‘thinly’ and ‘thickly’ specified work can also usefully be applied to time-based media installations. (Laurenson 2006, 5)

Here, Laurenson’s argument applies at the installation level, whilst Dourish’s applies at the level of code.

To evaluate the significance of an artwork’s materiality, Appelbaum’s concept of the value system (Appelbaum 2007, 89) – based on Riegl (1903) – can be applied. Although the system was originally developed for built heritage, it can be transferred to our discussion of software-based art. In particular, the historical value, newness value, art value, use value, and aesthetic value of such art are all relevant for a balanced judgement of the significance of digital materiality for preservation. Dourish’s concept of digital materiality, in combination with Appelbaum’s value system, can therefore inform decisions about whether or not software should be considered a significant property to be preserved.

3.5 Significant Properties in the Technosystem and Beyond

Significant properties serve as a base for the selection of suitable preservation strategies, and as indicators by which an intervention’s success can be measured. They can be compared to environmental indicators, tools used to understand and potentially steer the impacts of human activity on the ecosystem. Like these indicators, an artwork’s significant properties do not constitute a complete description of the artwork, hence why either a reference object or audio-visual documentation is also very useful for conservators. In contrast to environmental indicators, significant properties are often qualitative, and often cannot be assessed in a binary way (as either simply fulfilled or lacking). An example of a qualitative property is an artwork’s behaviour or concept, as these cannot be exclusively expressed in numbers. This pinpointing of specific properties for the purposes of quality control also makes sense given that it would be impossible to test all of the possible input values and paths of a software (Kraner 1997, 2).

Feenberg’s technosystem does not affect the concept of significant properties as described in section 2.3. However, it does offer a useful explanation of why significant properties are not a fixed set of parameters and values. As Feenberg sees it, the artist physically and mentally extracts the material and parts needed to create an artefact from their previous contexts, and creates new relations and connections between them. He calls this process decontextualisation. Then, when using the technical object, the user carries out

the opposite operation, and assigns cultural value to it. As such, the way in which the user uses the object is culturally determined. Feenberg terms this process interpretation or abstraction. As he explains, “The functional object is removed from its natural context, stripped of its useless qualities, and associated with other objects. Simultaneously it is interpreted in its new meaning and integrated into a cultural system from which it receives the ethical and aesthetic standards of the world it enters” (Feenberg 2017, 154).

Like the art creation process, restoration – or conservation – also has an interpretative character. As mentioned above, Groys writes that a conservator must decide (or interpret) whether the restitution of an artwork’s identity is based on its material continuity or form (Groys 1996, 156). In the example of the website and online performance *TV Bot* (2004), discussed previously, Marc Lee replaced hundreds of news channels with Twitter feeds and significantly changed the website’s design (see Figure 3 to Figure 5). Would it be acceptable for a conservator to effect such far-reaching changes? How often would such decisions have to be revised and conservation interventions carried out? When the artwork was created in 2004, visitors were fascinated by the fact that *TV Bot* provided more current affairs coverage than their TVs at home. However, this fascination has faded, as constant access to rolling news via social media and the internet has become the norm. As this experience of surprise cannot be preserved, it is therefore legitimate to ask whether it is really necessary to transform the artwork every few years to keep its functionality. With the introduction of the term sustainability, I suggest stabilising an artwork’s significant properties, even if this is considered to freeze it in a single state. This has the multiple advantages of reducing the scale and amount of changes made to the artwork, removing the need to select new conservation strategies with every conservation cycle, and simplifying the documentation effort. Thus, in the long term, the artwork has a better chance of remaining accessible. This does not mean that the concept of a structure with a permeable core, as described by Castriota (see section 2.3), is wholly inapplicable. If there is a new technology that could affect the preservation of an artwork, or if there is new knowledge about the work, its significant properties should be reviewed. Technical artworks often go through a process of development soon after acquisition, in which the artist and museum work together to stabilise the piece for display. The importance of such processes should not be underestimated. In science and technology studies, when a scientific problem is resolved through a consensus between those involved in studying it, this is called closure (Bijker 1995, 85). Therefore, an agreement between artist and institution about the significant properties of a software-based artwork could also be called closure, as here too we are

dealing with a process that begins when a new invention or scientific discovery is made. If closure is not possible, then having more than one set of significant properties for different preservation versions of the work becomes a possibility.

To summarise, whilst the technosystem offers a useful model for explaining the variability of an artwork's significant properties, frequently re-evaluating and redefining these properties makes securing consistent and efficient conservation strategies difficult. To improve long-term preservation, institutions should seek to identify – and agree on – a stable core of the artwork's properties in the medium term.

3.6 Preservation Strategies in the Technosystem

Feenberg articulates a description of the creative production of artefacts that is not only relevant to the question of reinterpretation, but also to other preservation strategies. "Something new comes into being through all technical activity and that is only possible because the subject projects itself beyond the present" (Feenberg 2017, 157). Although conservators try to preserve artworks and not create new ones, they have to try to imagine the future, in both technical and social terms. This may involve asking questions such as who the audience will be, how the artworks will be exhibited, and how people will experience them. Feenberg offers an account of the emergence of innovation as "the discovery of new potentials not included in the original design. Innovation depends on engagement with technical means which opens a range of initiatives that can be enlarged by a new perception of the object and its context. I will call this operation "abduction," in the sense of C. S. Peirce" (Feenberg 2017, 177). C.S. Peirce referred to abduction to explain the scientific method of knowledge production: abduction is the process of generating a new hypothesis from an unexpected observation. By way of analogy, Feenberg infers that the designer – through discovering an unexpected technical affordance – invents a new design that strives for this affordance or usage (Feenberg 2017, 182).

Abduction is a strategy that artists use to create artworks, but it can also apply to the reinterpretation of a work by imagining it in a different context. Furthermore, conservators often use tools or materials in ways other than what they were designed for. For instance, digital forensic tools were designed to identify crimes in the context of law enforcement, but are now also used to secure, analyse, and archive the digital content of software-based artworks (Kirschenbaum, Ovenden, and Redwine 2010, 1). Another example is emulation and virtualisation. The emulators and virtual machines that time-based media conservators

and archivists use were not intended for art conservation and presentation. Rather, emulators are often programmed by gaming enthusiasts to enable them to play video games or run obsolete software. Virtual machines emerged in the business context to enable more efficient, independent uses of hardware, and are part of the cloud infrastructure. On one hand, emulators and virtual machines help to access digital heritage, whilst on the other, because conservators do not need an intimate knowledge of the ins and outs of the software and hardware to be conserved, they therefore fulfil a bridging function. Further implications of the technosystem for preservation strategies will be discussed in the case studies, discussion, and conclusion of this thesis.

Preservation strategies must be able to accommodate the fact that "Durable physical artifacts project into the future the socially-constructed characteristics acquired in the past when they were designed. This is analogous to the persistence of acquired characteristics in a changing environment" (Hughes 1987, 38). In other words, although an artwork's environment evolves constantly, it retains its characteristics given at the time of creation. This applies to most artworks. However, for software-based artworks, the technological gap that opens between the moment of creation and acquisition, exhibition, or intervention quickly becomes significant for the way in which they function and are interpreted. Where there are external dependencies, this leads to dysfunctionality. As such, preservation strategies must bridge that growing gap, either by changing the artwork or adding an interface or bridge between the artwork and its environment (see Figure 12). The third option is to freeze the artwork within its closest environment, like protective padding. This approach still requires a bridge between the frozen environment and the larger, living environment around it. Usually, this is accompanied by a reduction in the number of dependencies.

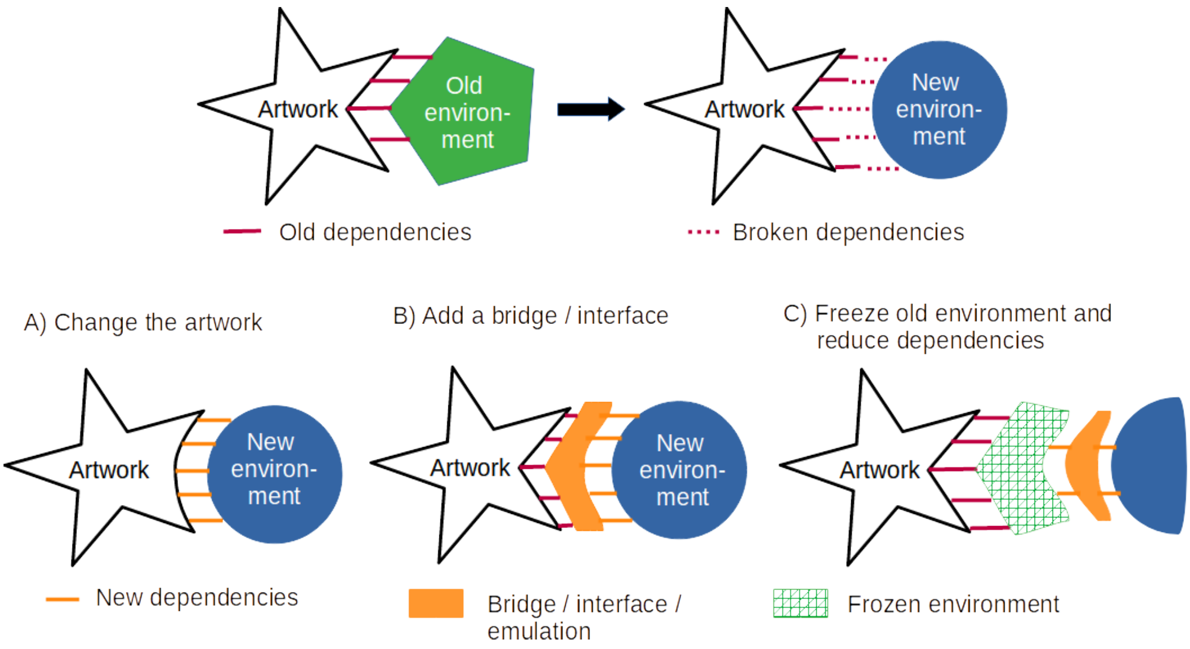


Figure 12: Principles of preservation strategies for software-based artworks.

Artworks are rarely designed with sustainability in mind. It therefore becomes the job of conservator and artist to make the artwork more durable. This can be a challenge requiring large interventions, similar to making a building earthquake-proof after it has been built. In addition to considerations involving conservation ethics, the sustainability criteria developed in section 3.1 should be applied to decide whether such measures are justified.

3.7 The Technosystem Applied: The Layers, Building Blocks, and Environment of a Software-Based Artwork

The technosystem describes an artefact's functional layers. In what follows, I extend this description to argue that understanding the functional layers of a technosystem – such as a software-based artwork – is a precondition for identifying sustainable preservation strategies. Functional layers exist in all digital systems, transforming and transporting data from input to output. Digital systems such as digital cameras or analogue/digital converters encode data such as video into binary code. Because binary code is not human-readable, it must be translated into a readable output via software such as a digital video player. This process can also unfold in reverse order: human-readable programming commands are compiled into machine-readable binary executables. This means that at least two layers can already be identified: the digitally encoded video and the audio-visual result, or the human-readable source code and the executable.

The following examples, illustrated in Figure 13, show that a pure software perspective is insufficient for grasping the processes of software-based art. The layers of software and hardware are always entangled. The two examples in Figure 13 represent very common situations: an output of a computer animation on a screen, and a sensor input to a software that produces an interactive experience.

In the first example, a game engine or multimedia platform outputs an animation in the form of a device-specific executable. Such an animation can only be interpreted within the right runtime environment, which consists of device drivers and libraries as intermediaries between hardware and software. In the example, a computer screen or projector is the output device for which a video card driver alone is sufficient. If other devices such as virtual reality headsets, game controllers, or speakers are used, their drivers would need to be installed as well. The operating system is the basic software layer which executes the system calls from the executable, and directs tasks to the video card instead of the central processing unit (CPU). The video card then outputs the video data stream through a standardised connector which is compatible with the screen or projector.

In the second example, the data flow is inverted: a sensor captures signals from the environment and converts them into digital data. This data is transmitted to the computer through a cable and standardised connectors. This raw data is then the input for an application that produces, for instance, an interactive animation. The operating system manages basic input/output operations and passes the data to the application, which is dependent on the sensor driver to process the sensor data. The application itself may consist of several layers or parts such as subroutines or program-specific libraries, and can interact with other pieces of software.

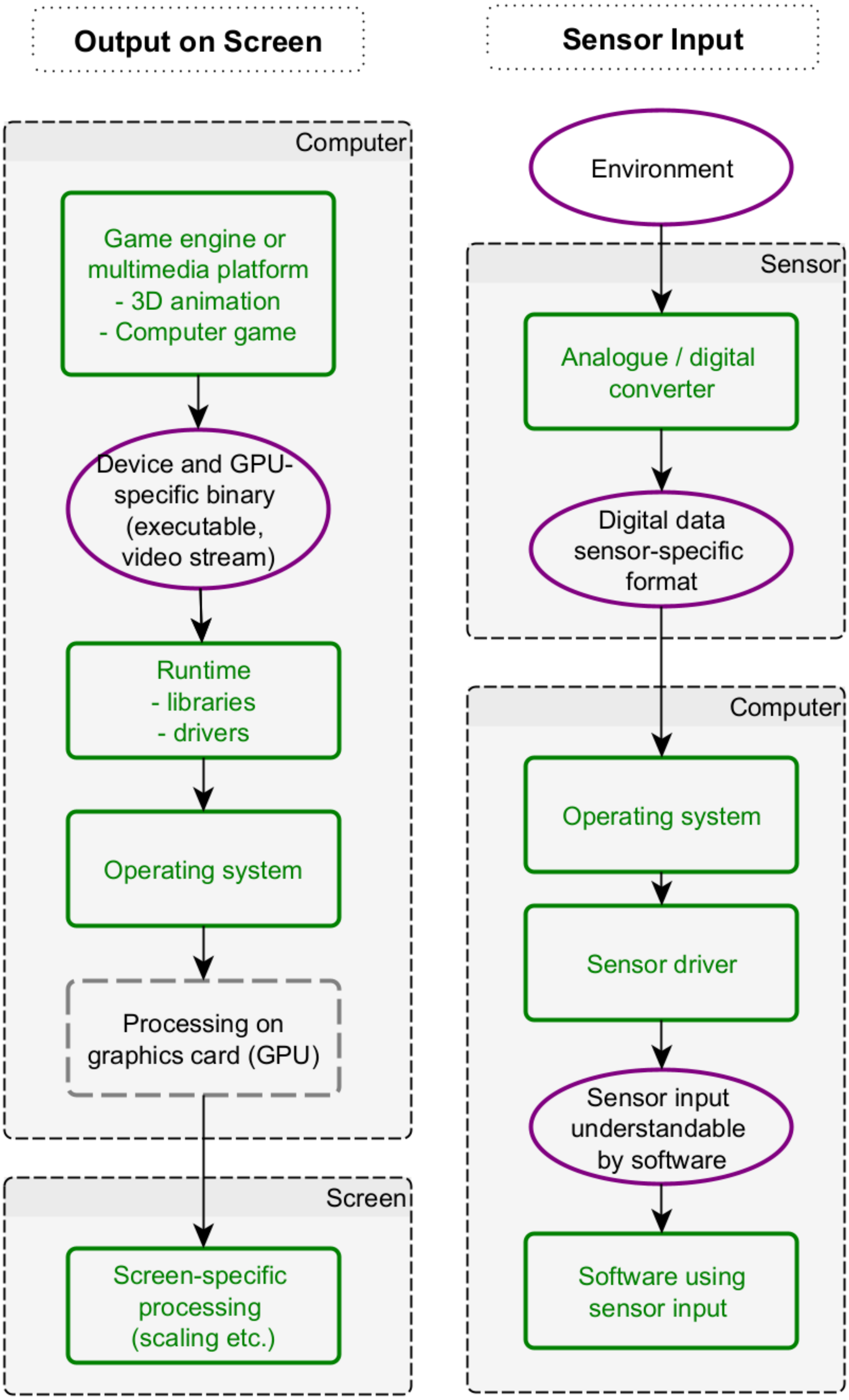


Figure 13: Example of output and input stacks, each consisting of a minimum of four software and hardware layers. Software and hardware layers in green, input and output in purple.

The software and hardware layers in the so-called software and hardware stacks are intertwined. According to Dourish, “The idea is that at each layer, the lower layer is both augmented (with new functions) and hidden (so that upper “layers” of the system need not worry about the details lower down)” (Dourish 2017, 59). If one level is removed or changed and the others left the same, the computation will most likely fail. These layers make it clear that the animation binary (software stack on the left) or the raw sensor data (software stack on the right) cannot simply be transferred to an arbitrary computing environment. Similarly, changing the input or output equipment requires adaptations to be made to the computing environment. Moreover, it is crucial to know which layers and building blocks have standardised interfaces so that they can be replaced or bridged more easily (as illustrated in Figure 12).

These software and hardware layers eventually determine an artwork’s behaviour and aesthetics, such as the speed and smoothness with which a pattern changes, the image and sound quality, or the reaction to inputs. All of these layers together generate the behaviour of the work. Indeed, it is often impossible to determine the contribution of any one layer in such dynamics (much like an ecosystem is more than just the sum of its parts). “One has to undo the separation between different ‘layers’ and ‘levels’ that computer scientists refer to as ‘the stack’ and recognize that the actual encounter with technology is not a serial encounter with each layer but a simultaneous encounter with a slice through the whole” (Dourish 2017, 151/190). Hence it is an advantage if the layers are kept together, by, for instance, running the software and its environment in an emulation or virtual machine, even if the individual layers are not a significant property and could be replaced or updated.

These virtual machines or emulators can be seen as individual computers with the software stack necessary to input and output data. These machines have different levels of dependency: the more dependent they are on the host machine, the more their software stacks are interconnected with it. There are also different levels of fidelity regarding software execution and outputs on a virtual machine compared to a physical machine, depending on the virtual machine’s level of abstraction. The more abstracted the virtual machine is from the hardware, the more independent it is from it, but the less precisely it emulates hardware-specific features.

The more functional layers are involved in an artwork’s performance, and the more they are interwoven and depend on each other, the more complex it is to preserve the artwork. Virtualisation and emulation enable conservators to preserve artworks without having to disentangle their layers. Paradoxically, such strategies achieve this by adding an

emulation layer, which serves to insulate the artwork and translate it over to the host environment, as well as replacing any dependencies on the original hardware.

The structuring of artworks into building blocks is also a highly effective approach to identifying preservation strategies. Often, artworks consist of several building blocks, each of which can be treated with its own conservation strategy. Each building block may contain several functional layers. A custom-made device driver might become dysfunctional if a new device replaces an old one. As a result, the driver will require updating or for some other solution to be found to make it work, and this strategy can differ from the preservation strategy applied to the artwork's other building blocks. A typical building block of web-based artworks is a client-side software stack, which can be approached using a different strategy to the server-side software stack. External dependencies, such as external data sources and formats, cloud-based software and platforms, and links to external websites, are all examples of possible building blocks; a computer or network node including software stack, or a large amount of similar data such as videos or images that require large amounts of storage, are others (see Figure 14). When it comes to identifying building blocks and appropriate conservation strategies for them, recognising dependencies is a prerequisite. An example of the significance of dependencies for preservation strategies is provided in Ilya Kreymer's article about the levels of website complexity and their implications for archiving strategy (Kreymer 2020). These levels of complexity are determined by dependencies.

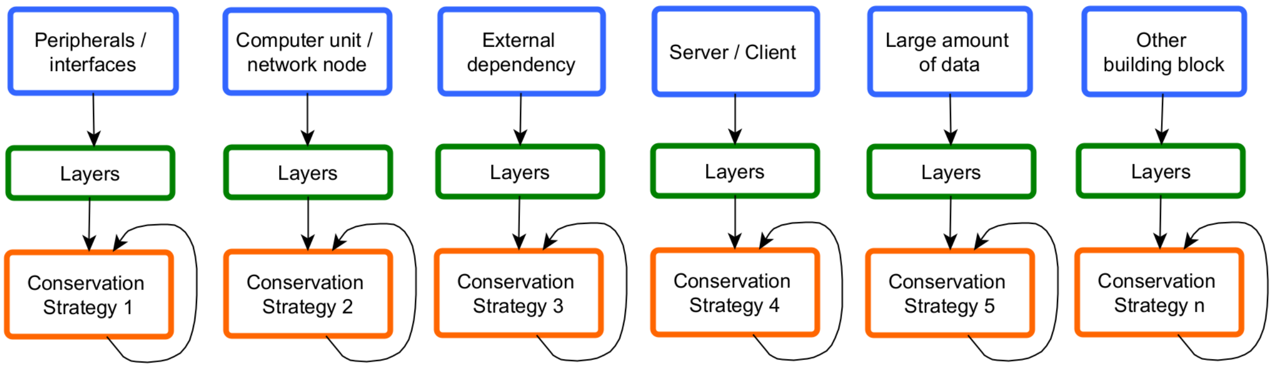


Figure 14: Examples of building blocks (blue boxes) and layers (green boxes). Each building block has its own conservation strategy (orange box). Identifying these strategies is iterative, and depends on whether the new preservation version protects the artwork's significant properties accurately enough.

Software-based artworks are highly dependent on and influenced by their environment. The software environment, such as the operating system and runtime environment, represents layers as described above, as does the hardware environment such

as the computer or the input and output devices. Replacing input and output devices with newer models for which there are no emulators usually requires adaptations to be made of the artwork in question. Depending on these devices' significance for the artwork, they and their software stack can be considered a part of it, or as its close environment, required for its functioning. The wider environment provides context and/or infrastructure for the artwork (see environment in Figure 12). For instance, web-based artworks often provide links to other contextual websites, which form part of the artwork environment. The boundary between artwork and environment is often blurred, however, and differentiating between an artwork's closer (to be preserved) and wider environment (either not to be preserved or preserved through documentation) can be useful. In any case, the environment usually requires its own conservation strategy and therefore becomes equivalent to a building block (see Figure 15).

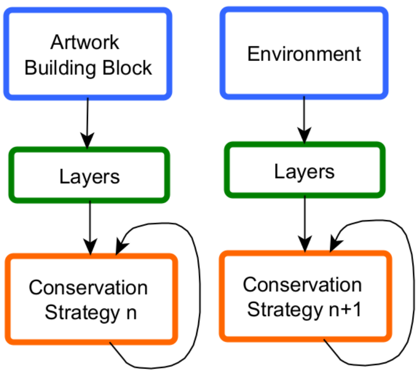


Figure 15: Building blocks, layers, and environment of a software-based artwork.

Distinguishing between an artwork's functional layers, building blocks, and environment is related to the software design principle of separation of concern. It encourages the development of independent preservation solutions for each of an artwork's parts, and allows for greater flexibility in maintaining and improving the conservation of these parts. Identifying preservation solutions requires knowledge of these technical layers. Although layers are co-dependent, they usually have a standardised way of communicating (for instance based on APIs, protocols). For this reason, layers are useful targets for the application of preservation strategies, for both an artwork building block and its environment.

3.8 Summary

The technosystem conceptualises the world and its subsystems as a field of technical practices that are associated with markets, institutions[20], and technologies. The introduction of the market as an influence within conservation theory is new. It is, however, a fundamental consideration for sustainable preservation strategies, as conservation decisions are highly dependent on the financial resources and conservation budgets of institutions. Sustainable preservation strategies must therefore strive to lower the preservation effort required for subsequent preservation cycles, and reduce the maintenance needs of artworks. Likewise, software complexity should not be increased, and external dependencies reduced. This lowers the risk of human error, and heightens the work's chances of remaining accessible many years into the future. The generalisation of components such as computers in the form of emulators is another way of reducing complexity and producing a certain scaling effect.

The technosystem assumes the ongoing development of society and technology. This assumption is important in order to understand not only the ageing process of software-based artworks, but also to see their conservation as a (slowed down, controlled) form of development, or an iterative process (see section 3.2). Sustainable preservation means both short-term and long-term thinking, and accepting the fact that software-based artworks will have to go through many preservation cycles during their life.

Dourish and Feenberg both provide a framework for seeing technical artefacts as not simply the product of their designer, but also of wider socio-technical and institutional processes. Until now, the artist's intent was often the primary criterion to consider in decision-making concerning conservation. The artwork's socio-technical environment was often taken for granted, and not explicitly addressed. This wider context, however, is vital for grasping the significance of an artwork's materiality.

Stabilising an artwork's significant properties for the sake of sustainable preservation seems to contradict recent trends in conservation theory which argue that variability in such properties is to be accepted. However, this stabilisation enables a more consistent application of strategies over the course of several preservation cycles, makes for less complex documentation, and reduces the risk of losing an overview of the artwork through multiple versions and interventions. In short, a balanced decision has to be made.

[20] In the context of this dissertation administrations as discussed by Feenberg are taken to mean institutions (see section 3.3).

The technosystem understands artefacts to be systems that are layered. Such layers have standardised interfaces. Software-based artworks, as one type of artefact, have several layers, each of which constitutes a potential target for preservation strategies. Dourish describes materiality as the simultaneous perception of all of an artwork's layers, like biting into a cake and tasting all of its layers at once (Dourish 2017, 151/190). As the contribution of a single layer to the overall behaviour cannot often be quantified, keeping layers together represents an advantage that emulation, as a preservation strategy, offers. Such an emulation layer not only protects the artwork from changes but also translates between the artwork's lowest layer (the operating system) and the host computer hardware.

In addition to identifying its functional layers, a software-based artwork should also be divided into building blocks, each of which can then be assigned an independent conservation strategy. Knowledge of an artwork's dependencies contributes to the identification of its building blocks, and hence of sustainable conservation strategies.

In the technosystem, artefacts exist in an environmental niche that they rely on to function properly. This perspective is also crucial for the preservation of software-based artworks. Indeed, most artworks of this kind are highly dependent on their environment. Ageing often manifests in the breaking of links between artworks and their environment, and not necessarily within the artwork itself. Furthermore, knowledge of an artwork's software environment is also essential for determining appropriate preservation strategies and identifying available options (see section 3.6).

4 Case Study 1: Evaluation of Preservation Strategies for *Horizons* (2008) and *Shan Shui* (2013) by Geert Mul[21]

This chapter introduces my first case study, the artworks *Horizons* (2008) and *Shan Shui* (2013) by the Dutch artist Geert Mul. Both works are based on the same software and practically the same hardware, but use different image databases. *Horizons* is owned by the Museum Boijmans Van Beuningen, whilst *Shan Shui* is owned by the artist. The two artworks are contained, which means that they do not rely on a web environment or an internet connection. They both employ mass-produced computers and equipment, are both complete (that is, they are in a finished state, and do not evolve over time), and are graphically intensive. As such, they are examples of interactive art installations that depend on a specific runtime (a combination of hardware and software) as illustrated in Figure 13, section 3.7.

Over the last ten years, much research into the preservation of digital heritage has focused on emulation. Indeed, the University of Freiburg researched and established Emulation-as-a-Service for memory institutions such as libraries and archives – a service that is currently also used for preserving software-based art. Meanwhile, software migration, which might provide an alternative solution, has been researched for its potential business applications, but less for software-based art. As an immediate strategy, migration has the advantage of not introducing an additional layer of translation, and thus not slowing down performance.

Sections 4.1 to 4.8 investigate the extent to which migration proves useful in the preservation of *Horizons*, and how this compares to emulation. The chapter as a whole addresses the question of the long-term impacts of migration and emulation on software-based artworks, the kinds of maintenance these strategies require, and what changes they

[21] Sections 4.1 to 4.8 and section 4.10 of this chapter are based on
Roeck, Claudia, Klaus Rechert, and Julia Noordegraaf. 2018. “Evaluation of Preservation Strategies for an Interactive, Software-Based Artwork with Complex Behavior Using the Case Study *Horizons* (2008) By Geert Mul.” In *IPres 2018*, edited by iPres 2018. https://osf.io/y3gcu and https://dare.uva.nl/search?identifier=4880bd3e-d3ba-4b16-9ead-a4ff77db2f0a
Section 4.9 is based on
Roeck, Claudia, and Teal Gaure. 2021a. “Wiki for the Documentation of the Permanent Installation of Shan Shui at the Dortmunder U.” Accessed November 24, 2023. https://gitlab.com/shanshui/boij/-/wikis/home.

induce in the artwork. As the case study shows, it was necessary to migrate *Horizons* before it could be virtualised with sufficient graphics to perform as intended.

Section 4.9 brings *Shan Shui* into this discussion of sustainable conservation strategies. In 2020, Geert Mul had the opportunity to install *Shan Shui* permanently at the Dortmunder U. For this, he had to migrate the software artefact to new hardware. A package manager in combination with a specific operating system was used to ensure that this migration would be reproducible.

Section 4.10 discusses the long-term effects of the conservation strategies applied to the two artworks.

4.1 Introduction

The fast obsolescence of hardware and software make continuous adaptations of a software-based artwork indispensable. The question addressed in this paper is what preservation strategies are more efficient or more sustainable in the long-term than others.

Geert Mul's artwork *Horizons* (2008) was chosen as a case study for this research, as its interactivity was difficult to describe and its video and sound were generated in real time, depending on a certain hardware. Geert Mul himself was in search of a long-term preservation strategy, as he comes across obsolescence issues each time, he has to reinstall one of his software-based works. *Horizons* seemed to have a setup which is not unique for this artwork and could be applied to other software-based works with a similar setup.

The comparison of two preservation versions carried out for the same artwork is chosen as a research approach in this paper. While the long-term effects of migration and emulation for the preservation of *Horizons* are discussed, their compliance with significant properties of the artwork are also investigated. As artists often do not provide any software specifications or significant properties, the identification of specifications for software-based artworks can be challenging. In order to complete or support the description of the significant properties, a method how to record computer in- and output is suggested. With this method at hand and with criteria for the sustainability of preservation strategies, preservation strategies and their combination can be evaluated. Hopefully, the long-term view on preservation strategies will facilitate future preservation actions.

4.2 Related Work and Definitions

Software as a component of contemporary artworks based on digital technology has been defined in various ways. This paper will either mention software as opposed to hardware. Or it will refer to software as a software artifact in contrast to its software environment such as an operating system. The Pericles project mentions a “digital environment“ and, due to its interrelationships, calls it a “digital ecosystem“ (Pericles n.d., 3). According to Osterweil software is “non-tangible, and non-physical, but often intended to manage tangibles“ (Osterweil 2008, 266). He stresses its hierarchical structure and that its components are interconnected and have different purposes. According to him, software requires modification and evolves. All these properties are essential and at the same time a challenge for its preservation.

These challenges can be tackled by means of different preservation strategies. In particular emulation was in the focus of the research about software preservation in the last few years. In 2008 Tabea Lurk and Jürgen Enge researched the use of virtualisation as a preservation strategy for software-based art in the sense of encapsulation of the digital object (Lurk 2008, 5). In 2010, Dirk Suchodeletz proposed a research agenda for the future integration of emulation (in the sense of encapsulation) into preservation workflows (Suchodoletz 2010). In the same year, Klaus Rechert, Dirk Suchodeletz and Randolph Welte suggested emulation as a web service (Rechert, Suchodoletz, and Welte 2010, 365), emulation also proved to be a practical strategy for exhibitions.

In the digital heritage community, migration is discussed rather for audio-visual components such as video tapes, film reels or audio tapes, or for text such as the migration from MS-Word-documents to pdfs, than for software-based art. The migration of software is a topic found more frequently in the business field. For instance, Wagner defines software migration as "splitting and transferring software to a new platform or technology (transformation) – with the goal to meet new requirements and to improve future maintainability. The existing functionality is to be preserved in order to prevent the loss of business knowledge." (Wagner 2014, 40) The focus of this definition lies on a smooth transition from the old to the new system and on enhancing functionality. For the preservation of software-based art, these requirements are not in the foreground, but rather the preservation of the artwork’s authenticity.

The JISC project (JISC 2008), carried out in 2006, defined migration of software as transferring a digital object from one software application to another. However, this

definition still reminds one of file migration. If the digital object is a complex software that consists of various files and file types that depend on each other, it requires a whole software setup and not just one single application to run them. Therefore, we define migration for this paper as follows:

Software migration means the transfer of a software artifact to an updated or different software environment in order to keep its functionality. This transfer can be related to new hardware, but it is not mandatory. It may include conversion (compilation) or adaption of the software to the new environment. The reprogramming of the software functionality with a different programming language is not considered migration in this paper. It is considered as a separate preservation strategy.

When comparing preservation strategies for software-based art, success criteria for long-term preservation are needed. The Library of Congress published sustainability factors for digital file formats which are, however, not completely suitable for software, as a software artifact fulfils a different function (active, computing) than a file format (passive, being processed). Patricia Falcao discussed risks that are inherent in software-based artworks (Falcão 2010), but she does not analyse what impact these risks have on the selection of preservation strategies. This is why this paper identifies success factors for long-term preservation and compares preservation strategies according to these criteria.

Comparing the ways in which different preservation strategies impact the functionality and appearance of software-based art requires the identification of the significant properties of the artwork. Significant properties are well known in the field of software engineering. Wilson defines significant properties in relation to digital preservation: "The characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record. Properties are considered to exist in one of five categories: content, context, appearance, structure and behaviour." (A. Wilson 2007, 8). Stephen Grace further develops the model in so far, that the different needs of stakeholders are taken into account (Grace 2009, 6). The model is applicable to software, but it is not broad enough for a software-based artwork as this can be based on specific spatial, temporal, and hardware context. Laurenson fills that gap and lists 15 areas of focus for significant properties for software-based art[22] including Wilson's and Grace's

[22] 15 Areas of Focus for Significant Properties of Software-Based Art: 1- Content and assets. 2- Appearance 3- Context. 4- Other Versions. 5- Formal and Structural Elements. 6- Behaviour. 7- Time (Durations of processes etc.). 8- What are the Spatial or Environmental Parameters of aWork? 9- External Links or

categories (Laurenson 2014, 92–94). These areas of focus served as a checklist for the definition of significant properties. As some of these areas of focus were overlapping and some were not relevant for *Horizons*, the significant properties have been summarized under "Idea of the work", "Processes implemented by the software", "Look and Feel" and "Hardware Dependencies" in the following section.

4.3 Significant Properties and Dependencies of *Horizons* (2008)

This section analyses the artwork *Horizons* (2008) by Geert Mul and determines its significant properties. It also provides the context of its acquisition and its preservation up to 2016 when this research started, as it had an impact on how I was able to determine *Horizons'* significant properties.

The Museum Boijmans Van Beuningen commissioned and exhibited *Horizons* in 2008. After its exhibition, Geert Mul bequeathed *Horizons* to the museum (Museum Boijmans van Beuningen n.d., 123). When Geert Mul prepared for his retrospective at the Stedelijk museum in Schiedam in 2016, he contacted the Museum Boijmans Van Beuningen in order discuss whether *Horizons* could be displayed at his retrospective. However, it turned out, that the Museum Boijmans Van Beuningen only had a documentation video of *Horizons*, which it took to be the artwork. The Museum Boijmans Van Beuningen could not find any confirmation that the artist had transferred the artwork to the museum. It appeared that the artist had not transferred *Horizons*, as it had caused problems in the exhibition in 2008 and he wanted to stabilise it first. As the museum did not have a budget to buy the hardware for *Horizons* in 2016, this research was done based on the artist's own instantiation of *Horizons*.

While I saw *Horizons* only on computer screens and not as an installation, I had the opportunity to accompany Geert Mul when he installed a similar work, *Shan Shui* (2013), in his retrospective[23] at the Stedelijk Museum Schiedam in 2016. *Shan Shui* uses the same software as *Horizons*, and almost the same hardware. The only difference is the image database they use and the aspect ratio of the projection. There, I could see how Geert Mul adjusted the installation parameters and what properties were important to him. A video[24]

Dependencies. 10- Function 11- Processes. 12- Artist's Documentation of Process (status of this documentation? Relationship to the work?). 13- Rules of Engagement. 14- Visitor Experience. 15- Legal Frameworks.

23 Exhibition *Match Maker* at the Stedelijk Museum Schiedam in 2016.

24 (Mul 2017a) Video made by LI-MA in 2017.

where he stands in front of *Shan Shui* explaining what the artworks means to him, as well as an artist interview[25] I conducted with him about *Horizons* (2008) and *Shan Shui* (2013) were other sources I used to determine the significant properties of *Horizons*.

The significant properties will serve as preservation objectives and benchmarks for the evaluation of the preservation strategies applied later. For the preservation of the work, several aspects have to be considered: the overall idea of the work, the processes that are implemented by the software including user interaction, the look and feel of the work which is a combination between software and hardware effects and the dependencies of the software on hardware. The latter has an impact on the preservation of the work. The spatial installation parameters will not be discussed here; although significant for the re-installation of the work, they are not relevant for the choice of preservation strategies.

Idea of the Work

The artwork *Horizons* by the Dutch artist Geert Mul is an interactive multimedia installation that was commissioned by the Museum Boijmans Van Beuningen in Rotterdam, The Netherlands. The work consists of reproductions of landscape paintings from the Boijmans Van Beuningen collection in an interactive installation. The horizons of these digitized landscape paintings are all aligned to the same height (s. Figure 16).

[25] (Mul 2017b).

Figure 16: *Horizons* by Geert Mul in the Museum Boijmans Van Beuningen in 2008 (source: website Geert Mul)

The concept of the work refers to the project *Search for a Horizon* by the Dutch artist Ger van Elk in 1999, also at the Museum Boijmans Van Beuningen, where the artist hung paintings from the Boijmans van Beuningen collection side by side, with their horizons aligned. In his installation *Horizons*, Geert Mul extended this idea by presenting the collection in an interactive, dynamic, playful way. He achieved this by developing a computer program that generates a video of the digitized paintings which is projected to the wall by three digital video projectors (s. Figure 17).

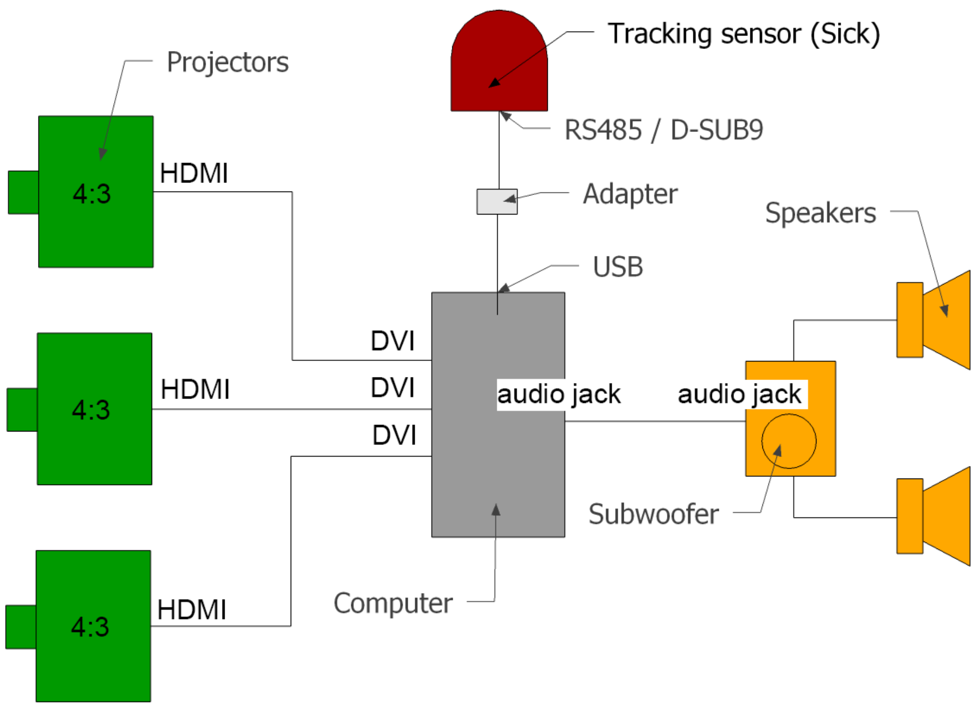


Figure 17: Equipment schematic for *Horizons* by Geert Mul (artwork creation 2008, version 2013)

Visitors walking around the space cause the image to split where they stand, revealing a new painting underneath. This interaction is enabled by a tracking sensor that sends the visitor's movement data to the computer. The sound pans seamlessly between right and left speaker depending on the visitor movement and on the number of visitors. The computer, projectors, sensor and speakers are industrial, mass produced, and they are either hidden from view or positioned in the background. Their physical appearance is not important for the work.

From this description can be concluded, that the interactivity and generative character of the video are significant for the preservation and that the equipment is replaceable, as long as its functionality is maintained. It can also be said that the software is used as a tool to achieve the effects of the interactivity and video and sound generation, but its significance for the artwork does not extend beyond that as long as its functionality is maintained.

Although the Museum Boijmans Van Beuningen commissioned *Horizons* for an exhibition on their premises, and although *Horizons* presents the landscape painting collection of the Museum Boijmans Van Beuningen, the artwork is not site-specific. In a

discussion[26] with Geert Mul and Sandra Kistler, the head of collection at the Museum Boijmans Van Beuningen, Geert Mul explained that he did not want the images to be exchanged when the work is shown at a different venue. According to him, the work is “strong enough” to be shown in a different context.

Processes Implemented by the Software

In a close collaboration with the artist, the programmer, Carlo Prelz, wrote the software from scratch. Prelz programmed the low-level code in C and compiled it into native machine code. He wrote the higher-level code in Ruby, which is an interpreted language. The lower-level code enables the communication with the tracking sensor and the interpretation of its location data. Additionally, it handles the image composition for the video. The C code is specific for this tracking sensor brand and technology, a time-of-flight laser sensor. The Ruby code generates the video and the reaction to the movements of the visitor.

When a new visitor enters the space, the computer randomly chooses a new image from the database. It also randomizes the size and orientation of the image. The visitor's horizontal position defines where the new painting appears on the screen: When the visitor moves parallel to the screen, he or she drags along the painting. The movement of the painting is accompanied by vertical ripples running across the screen, similar to a stone falling in the water and causing circular ripples. The visitor's distance from the screen determines the height of the projected painting. If the visitor moves towards the screen, the painting shrinks. If the visitor stops or leaves, the ripples continue to run towards the left and right border of the image until they are completely gone. Without a visitor moving, no new ripples will be triggered, and the image is still. When a new visitor enters the space and stands in front of the projection, the algorithm randomly chooses a new sound sample. The horizontal position of the visitor (movement parallel to screen) shifts the volume between left and right speakers. The vertical position of the visitor (movement away from the screen) increases the pitch.

This description of the processes is neither precise nor complete. As the generated video is the product of several programs running at the same time (reading sensor input, random image selection, creation of image columns, reaction to sensor input, etc.), the behavior of the work is difficult to predict. Without following the exact algorithm, the

[26] (Mul, Zijlstra, and Kisters 2017).

processes cannot be described in a precise and complete manner. As Rothenberg already indicated in 1999:

> "[...] we do not yet have any formal (or even informal) way of describing the full range of behaviors possible for even the simplest of digital documents, such as are produced by word processing programs. Describing the behavior of dynamic, interactive, hypermedia documents poses a far greater challenge. The only adequate specification of the behavior of a digital document is the one implicit in its interaction with its software. The only way to recreate the behavior of a digital document is to run its original software." (Rothenberg 1999, 22)

The behaviour caused by a complex program can have infinite variations and be unpredictable. It is not possible to prove that such a program is complete and correct.

As a consequence, although the software (Ruby and C-Code) is only used as a tool in this artwork and thus in itself is not conceptually relevant, the source code is regarded as a significant property of the work as the behaviour it causes through its execution is too complex to describe and thus hard to fully replicate in other code.

Look and Feel

The look and feel of the video is equally difficult to describe. Deducing from the source code, the rate of the video generation is dependent on the image size, the CPU speed and the video card. As a consequence, it is important to determine the speed of the video patterns via visual observation. The image columns (ripples) triggered by the visitor move across half of the screen in about 15 to 30 seconds, which corresponds to a horizontal movement of about 50 to 100 pixels per second, very roughly estimated. For the artist it is important that the movements in the video are smooth[27].

In addition to its projection speed, the video quality is also a consequence of the quality of the images in the database. The colour space, deduced from these images, is RGB, 8 bit per channel, the resolution of the video frame 3072x768. The artist considers the video resolution as a significant property, although he would not exclude upscaling if the current type of projector cannot be used and purchased anymore.

The sound quality is determined by the quality of the 13 wav-sound samples, which are stereo, have a bit-depth of 16 bit and a sampling frequency of 44 kHz. The sound

[27] Oral communication with Geert Mul during the installation of *Shan Shui* (2013) in the week of 28th of October until the 4th of November 2016.

resembles a pink noise. As a precise description of the look and feel is impossible, a video and sound recording can serve as a reference documentation.

Hardware Dependencies

In 2017, Geert Mul supplied a standard desktop computer with *Horizons* installed on it to LI-MA (LI-MA n.d.b), a platform for preservation, distribution and research of media art, for research purposes. The Intel-based x86/64bit computer was equipped with two built-in Nvidia video cards with each two DVI connectors, and several USB sockets (s. Figure 17). As most files on this computer were created in 2013 and the components of the computer date from 2012/2013, this version will be called version 2013 and serves as a reference. This version is dependent on the Nvidia video card, as this type of video card is hard-coded in the source code of the artwork (s. section 4.6, "Analysis of dependencies (Version 2013)).

For the first version of *Horizons* in 2008, Geert Mul used an infrared camera as a tracking sensor. While the camera was very expensive, it did not work well. They had to use a parabolic mirror, as the thermal camera was not wide angle enough. In addition, the camera had difficulties to recognise a person at different room temperatures, which is why they had to recalibrate the camera from time to time (interview with C. Prelz, 2017b, May 21). Thus, Geert Mul tried to find a stabler solution for the tracking sensor. A friend of Geert Mul, Marnix de Nijs, lent him a sensor based on a laser beam, and Carlo Prelz, the programmer, adapted the program for this sensor (probably around 2013, [interview with C. Prelz, 2017a, March 12]). This setup worked reliably and was used for the artwork described in the following sections.

Thus, since around 2013, Geert Mul is using a time-of-flight laser sensor, LMS 200 SICK, which will be called SICK sensor from here onwards. Although this is a sturdy industrial sensor with very basic maintenance requirements, it has to be assumed that this sensor will have to be replaced in the future. As there is no standard protocol for tracking sensors, the driver of the sensor sick.c was taken over from the media art lab V2, Rotterdam, adapted by the programmer and integrated in the artifact. Thus, the sensor cannot be replaced without additional programming measures.

Comments Regarding the Definition of Significant Properties

The definition of significant properties of an artwork can be contestable; often, they are not measurable or technology dependent qualities that are not applicable anymore when

the work is transferred to a different technology. Even if one manages to quantify a property, it is difficult to know how big the scope of acceptable deviation is - an aspect that the artist him/herself may also be unable to quantify. Besides, the definition of significant properties is partly dependent on the values of the stakeholders. For instance, a team of curators and conservators might emphasize aesthetic values above the historical values of the work, or the other way round. Grace also recognizes, that there is not one single, definitive interpretation of significance (Grace 2009, 6). Bettivia confirms this in her research about the significant properties of a video game where she described that different designated communities defined different significant properties for the same game (Bettivia 2016a, 213). Finally, the perception of what the significant properties are can shift with each new exhibition, development of technologies and of conservation theory.

The subjectivity and possible shifting of the definition of significant properties can be countered by keeping deprecated artwork instantiations, their documentation and description of significant properties. By keeping the history of the definitions of significant properties transparent, new conservators and curators can make informed decisions about what they consider significant.

Furthermore, if significant properties are either incomplete or hard to verify - as this is the case for *Horizons* - and therefore insufficient for the application of a successful long-term preservation strategy, a viable solution is to keep as much of the "original" as possible, which was aimed at for *Horizons*. Finally, the mixed approach of technology independent description (description of function, interaction, behaviour and processes) and technology dependent properties (colour space, video and sound resolution, source code, hardware dependencies) comprises a wide range of properties and therefore supports long-term preservation.

4.4 Considerations for Long-Term Preservation

In order to enable sustainable preservation decisions, not only the artwork's significant properties, but also criteria for a successful long-term preservation are needed. This will enable the comparison of preservation strategies. The following paragraphs establish these comparison criteria.

Adaptability to new hardware. Hardware abstraction is an important means to facilitate future preservation measures. The more independent the software is from the hardware, the easier it will be in the future to transfer it to new hardware. Even if the

transfer of an obsolete software to new hardware is successful, this will most likely require software changes.

Resilience to software obsolescence and adaptability to changed network protocols. The resilience towards obsolescence of software applications and programming languages is therefore another important factor for the long-term preservation of software-based artworks. As any hardware change can lead to a change of software requirements and/or configuration, preservation measures that prepare the artwork for or shield it from such changes support long-term preservation. For web-based artworks, the abstraction of protocols is relevant to achieve more independence from changing Internet and data protocols.

Stabilization of software complexity and minimizing change rate. Although for the preservation of artworks usually no new features are added, it might be necessary to adapt the software to new hardware. The more changes a software undergoes, the higher is the risk that bugs or deviations from the significant properties are introduced. Preservation measures should therefore try to stabilize the software artefact and reduce its change rate.

Ease of installation of the artwork / of connecting peripherals. Ultimately, the goal of preservation is the presentation of the artwork. If the installation is very fragile or if it is difficult to find the right settings in order to adjust it to the space, the risk is higher that it will not match the significant properties. Furthermore, if it is very complex to install, the museum might be inclined to display it less, as the installation costs are high. For these reasons, the ease of installation and the stability of the installation process can be considered success factors for long-term preservation.

Ease of maintenance and of function tests (monitoring). Article 17 of the E.C.C.O. guidelines (II) mentions that the conservator-restorer should specify the most appropriate means of continued care. Maintenance is another, more technical term for this and is essential for the continuation of technology-based art. This is also valid for software-based art, as Lehman and Ramil describe when explaining software evolution: " In general, the change will be such as to adapt the elements of the class so that they maintain or improve their fitness to a changing environment." (M. M. Lehman and Ramil 2002, 275). The aging of software-based art is not inherent in the software, but rather in the interdependencies of software with hardware and external data sources and protocols. The external environment of the software-based artwork will continue to change, even if the software artifact itself does not. This process will eventually lead to its dysfunction, if no maintenance is undertaken. Maintenance prepares the artwork to be installed at any given time without

having to start a restoration project first. Through these small maintenance activities, the artwork's functionality is automatically monitored and big surprises regarding its functionality are prevented. Ease of maintenance and of function tests can be expressed in time spent on these maintenance activities each year, taking into account the time saved for large-scale restoration work.

Scalability of preservation strategy. Since tailor-made approaches to the preservation of software-based artworks are time consuming and expensive, more generic solutions such as emulation or virtualisation are preferred. Specific hardware requirements might prevent this approach, but some artworks might lend themselves to a more generic approach. This would facilitate their preservation and change management. Just a few emulators would have to be maintained, instead of maintaining and migrating many different software environments. The cost of the emulator maintenance could be shared between the artworks.

4.5 Preservation Options for *Horizons*

The following sections discuss the digital preservation strategies available for *Horizons* and give reasons why a strategy was chosen or rejected for its subsequent execution. Practical arguments of feasibility and available resources were taken into consideration.

Reprogramming

One preservation option is, to reprogram a software-based artwork in a different programming language and for a current software environment. The reprogramming can either be based on formal software specifications that again are based on the significant properties of the artwork or on formal specifications such as pseudo code, a generalized interpretation of the source code.

The Guggenheim museum provides an interesting example of a reprogrammed work: it reprogrammed parts of *Brandon* (Cheang since 1998), a web-based artwork created in 1998 by Shu Lea Cheang, by directly translating the existing non-functional code into another language; all the Java applets were replaced with GIFs, JavaScript and new HTML (Phillips et al. 2017). While this measure (in combination with other measures) has the immediate effect of making the artwork operational again, this huge effort will have to be repeated when one of the technologies used becomes outdated.

Reprogramming based on software-specifications requires an exact description of the significant properties of the artwork. In the case of *Horizons*, where several software components interact with each other to produce the effect on the screen, the exact description of the behaviour is not possible. Even though complex behaviour can be caused by simple functions, these functions can be difficult to recognize. To re-engineer such behaviour can therefore require a very substantial effort and the result might not be sufficient.

Reprogramming based on pseudo code would be a very interesting option, as pseudo code could be preserved as text in the long-term without problems. Rinehart who compares the source code of an artwork with a musical score suggests: "A system of formal notation for media art should be abstracted from specific environmental factors. It should be robust, generic, adaptable and portable - universal content for a universal machine." (Rinehart 2007, 182). While pseudo code could theoretically adopt the role of such a generic, portable code, the question has to be asked whether pseudo code and the subsequent translation to a new programming language can express all idiosyncrasies of the original language. Finally, for *Horizons*, translating several thousand lines of software into pseudocode and then reprogramming it in a different language was considered too expensive. These reasons were the reasons why this strategy was renounced.

Migration

Moving and integrating a software artifact into a different software environment without having to reprogram it is referred to migration in this paper (s. section 4.2). Migration may be necessary due to the updating or complete change of the operating system, or due the replacement of important libraries. They are typically a consequence of the transfer to new hardware. Even though changes in the hardware or software environment may require a recompilation of the source code (if available), this preservation strategy is based on the same source code or binaries and thus, the same algorithms as used for the original work.

The migration of source code was chosen as the easiest and most straight forward migration option. It was also financially feasible and still interesting regarding the effect of the recompilation on the installed work. The operating system Debian was not replaced by a different one, as it is open-source and quite common. However, it was chosen to update the operating system from Debian 7 to Debian 9 in order to use a current system and in order to

test the effects this has on the significant properties of the artwork. Finally, it was a goal of this research to analyse the migration process.

Emulation

An emulator is able to interpret the machine language commands of a specific CPU type or computer architecture. In practice emulators represent a complete computer system, typically including a sound card, graphics or network adapters. Classic virtualisation is reverted to when the performance of the emulation is not sufficient. In contrast to emulation, the guest system has direct access to the host computer's CPU and the system commands do not have to be translated form guest to host.

The emulation and virtualisation were both chosen as one strategy, as they can be executed with the same emulation/virtualisation software but with different settings. Although there were concerns regarding the performance and video card dependency, it was decided to test this strategy, as emulation offers the highest abstraction level of all the options mentioned and therefore interesting long-term preservation perspectives.

4.6 Carried out Preservation Strategies for the Software

Analysis of Dependencies (*Horizons* Version 2013)

As there was no documentation of the software, some detective work had to be undertaken in order to be able to isolate the artefact from its software environment and in order to determine and document its hard- and software dependencies. As a preliminary measure, disk images were taken from the two internal hard drives of the artist's computer.

The disk image enabled the access to the source code and the identification of the folder structure of the project. From this analysis and interviews with the programmer Carlo Prelz[28], it appeared that the same software was used for several artworks and projects. The programmer made many changes to the software, which resulted in unused source code files located in the same project folder as the *Horizons* project.

Another resource of analysis were error messages on the artist's computer. When the sensor was not connected, the program started and an error message pointing at a central Ruby program popped up. Starting from that program and following the calls it made for other programs and libraries it was possible to understand the start procedure of *Horizons*.

---

[28] (Prelz 2017a) and (Prelz 2017b).

The same method of source code analysis was used to understand which files were part of the *Horizons* software and which libraries it depended on.

The source code analysis showed that C-programs were wrapped and executed through Ruby. To integrate C-code into Ruby the programmer compiled the C-code as a library that can be used by Ruby. For *Horizons*, this Ruby library is called boij.so (s. Figure 18). It integrates amongst others a C-program sick.c, which constitutes a manually written driver for the SICK tracking sensor. Furthermore, the programmer added custom-built and non-standard libraries to the standard C- and Ruby libraries. To know their location and the location and version of the compiler and interpreter used is indispensable for a migration.

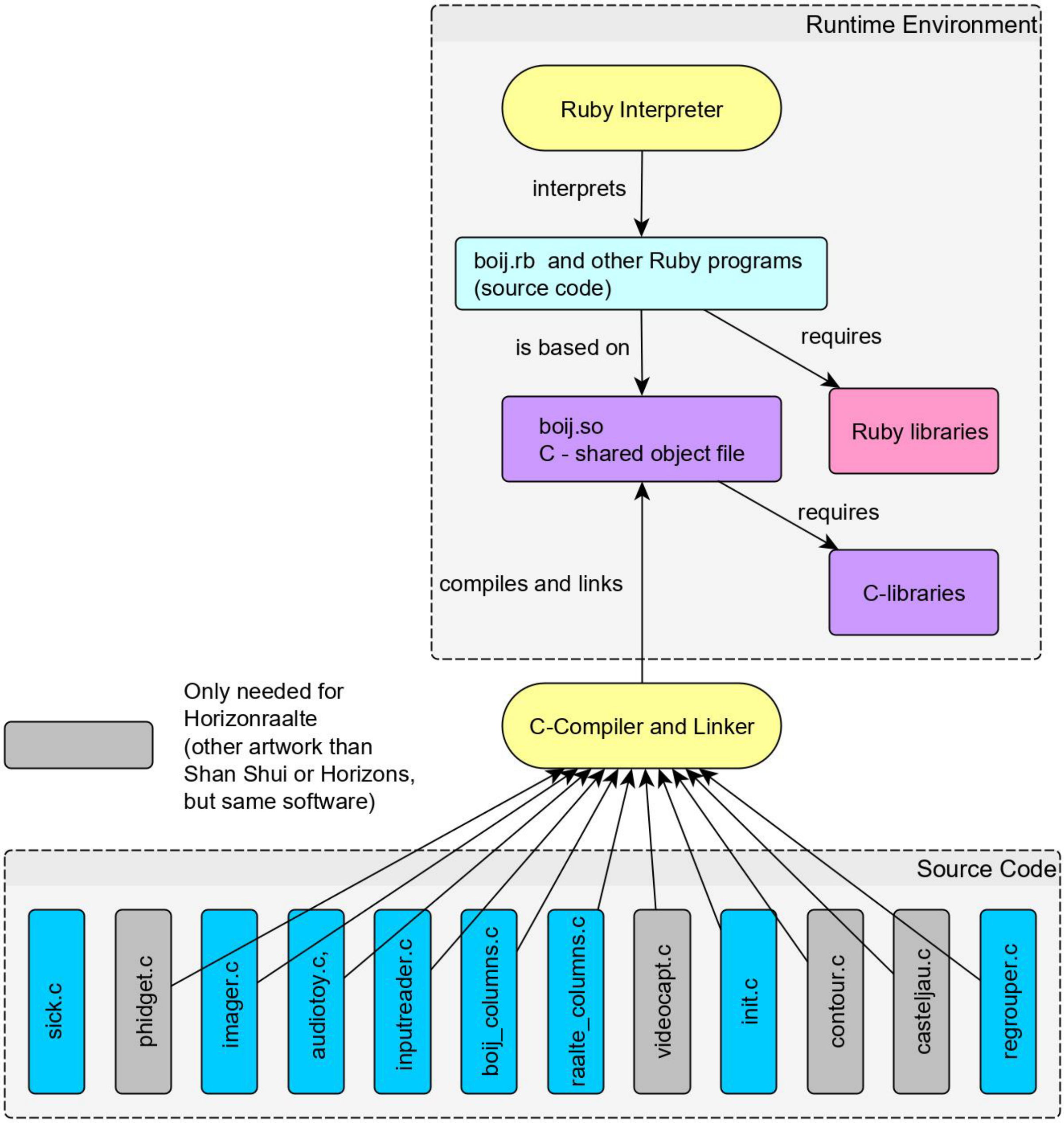


Figure 18: Simplified schema for runtime environment and source code of *Horizons*

A further result of source code analysis was that intermediate hardware abstraction layers such as SDL and OpenGL were used in order to generalize the underlying hardware. Nevertheless, the model of the video card was hard-coded in one of the Ruby programs and in the configuration file of the display manager. The use of the hardware abstraction layers and the fact, that the SICK sensor was connected to the computer through a standard USB interface was the reason, why an emulation approach was considerable.

A set of 113 images with varying aspect ratio and resolution are used for the video generation. The programmer converted these images previously from the PNG-format to a project-specific bo2 -format with the Ruby library RMagick. An image database (SQLite) contains a list of IDs of the images used for *Horizons* and the position of their horizons. The sound generation of *Horizons* is based on 13 sound files in WAV-format. The knowledge of these file locations and relations is relevant for a migration of *Horizons*.

The configuration file for the SICK sensor was in binary format and was overwritten each time, the SICK sensor was re-connected to the computer. It contains site specific sensor and software parameters. Thus, the parameters of past exhibitions were neither saved nor were they human readable. From the preservation point of view, this is a lack of the software which could be amended when migrating it (creating a new preservation feature).

Although the programmer installed Debian 7 as on operating system, he integrated an old Linux kernel (2.6) in order to be able to use the LILO boot procedure which he preferred to GRUB, the new standard boot loader for Linux kernels. Based on that boot procedure *Horizons* started up as soon as the computer was switched on and shut down in a controlled way, when the computer was switched off. This boot procedure has to be known when migrating the work.

Migration Version 2017

The migration of *Horizons* not only comprised the update of the operating system, but also the implementation of several other measures supporting the preservation of *Horizons*. In particular, it was a goal to make the work independent from a specific video card by detecting the video card automatically, so that it could be presented on any computer (with x86/64bit architecture and a capable video card).

The new base-system was built from scratch based on the knowledge gained from the runtime analysis described above and in Figure 18. The most current operating system (Debian 9 / Linux kernel 4.9) was installed in order to have a stable version that is able to deal with new computer hardware and to avoid having to update soon after the migration

(OS of the 2013 version: Debian 7, kernel 2.6). This operating system update also included a new version of the Ruby interpreter and the C-compiler.

As a next step, the source code, the images, the image database and the sound were copied over. Not all data in the artist's data folder were transferred, as some directories contained other projects that were not directly related to *Horizons*. The separation and deletion of unrelated data is an ongoing task that has not been finished. Cleaning up decreases maintenance and increases the understanding of future conservators. However, throwing out seemingly unrelated data does entail a risk of breaking the code and of deleting the project history. This is why it is important to use a version control system for the source code, a solution that can support the principle of reversibility that is central to arts conservation.

After the transfer of the source code its C-library was recompiled, to match updated ABIs[29] of the new system. The Ruby programs did not have to be recompiled, as they are interpreted on the go.

For this migration, the tracking sensor was not replaced by a new model. Due to the analysis of the software in Figure 18 it is clear, though, that the sick.c file (driver of tracking sensor) would have to be adapted and recompiled should the sensor be replaced.

The procedure to boot the *Horizons* version 2013 was based on the LILO boot loader. As the development of the LILO boot loader ended in 2015, the automated start-up procedure of *Horizons* had to be adapted to use the GRUB bootloader. The change of the automated start-up procedure also included the configuration of a new display manager (NODM) to invoke the graphics window for the video.

This step-by-step reconstruction of the new system indicates that, to a certain extent, it was possible to separate the software artefact from its software environment. This knowledge is useful for future migrations and the technical understanding of the work as it will be possible to establish whether the replacement of certain software components (libraries, operating systems) is possible.

Emulation / Virtualisation Version 2017

Emulation (virtualisation) was chosen as a second preservation strategy as it makes *Horizons* more independent from hardware than migration. The original goal was, therefore,

---

[29] Application Binary Interface.

not to change the software and its environment by running a disk image of the internal hard drive of the 2013 version in the emulator.

Graphic rendering, as used in *Horizons*, is computationally intensive and was therefore offloaded to specialized hardware, a graphics processing unit (GPU) usually part of high-end video cards, to increase the rendering performance. In theory, GPUs can be emulated, i.e., the functionality of GPUs can be implemented in software which is then executed on a generic CPU. GPUs provide highly specialized functionality, optimized for specific tasks, such that a generic GPU emulation - even for older GPU models - provide a poor performance in practice. We tried to run the emulated setup using software rendering only, but the graphics rendering was not sufficient. In order to provide GPU functionality for the emulated / virtualized environment a different approach is necessary.

Emulation options for the graphics rendering. QEMU was chosen to emulate (virtualize) *Horizons*. QEMU and similar emulation and virtualisation system are able to use the host's CPU to execute binary code directly (CPU virtualisation) for performance reasons. For preservation purposes, virtualisation is a technology to bridge the gap until either CPUs are fast enough to emulate even contemporary software or the current architecture changes significantly, such direct execution becomes impossible.

A similar approach for GPUs is required. One potential option is a GPU pass-trough, as used by public Cloud providers today[30]. With GPU pass-through, the original vendor driver is installed in the guest system, which however, ties the virtualised instance again to a specific hardware setup. An alternative is to install a generic virtual GPU within the guest system and delegate the adaption to the host's GPU hardware to the emulator implementation. The Virgil3D GPU project added such functionality to the QEMU emulator recently (VirGL community since 2018). When using virtual hardware, a new guest driver has to be implemented (in contrast to emulated hardware for which the original vendor drivers could be used). On Linux systems, this driver is available since kernel version 4.4[31]. As the Linux kernel of the artist's version was 2.6, the kernel had to be updated. Instead of only updating the kernel, we decided to use the migrated version 2017, as it already contained an updated kernel (Debian 9.1 including a Linux kernel 4.9). This allowed a direct comparison of the same software environment running on real hardware with a virtualised instance. Thus, not the 2013 version of *Horizons*, but the migrated 2017 version of *Horizons*

[30] E.g. Announcing GPUs for Google Cloud Platform, (Barrus 2016).
[31] A driver for Windows guest is currently under development (Gauër since 2017).

was emulated (virtualized). In other terms, the virtualisation contains the migrated version in the guest system. Figure 19 illustrates the virtualised system setup of *Horizons*. The footnote[32] below gives the QEMU command that is necessary to start the virtual machine.

[32] qemu command line for *Horizons* with accelerated graphics rendering: QEMU_AUDIO_DRV=pa qemu-system-x86_64 -enable-kvm -cpu host -m 4096 -vga virtio -display gtk,gl=on -machine smm=off -usbdevice tablet -netdev user,id=net0,hostfwd=tcp::22222-:22 -device e1000,netdev=net0 -soundhw hda -drive if=virtio,file=\$dir/shan-shui.qcow2,cache=none -k en-us -usb

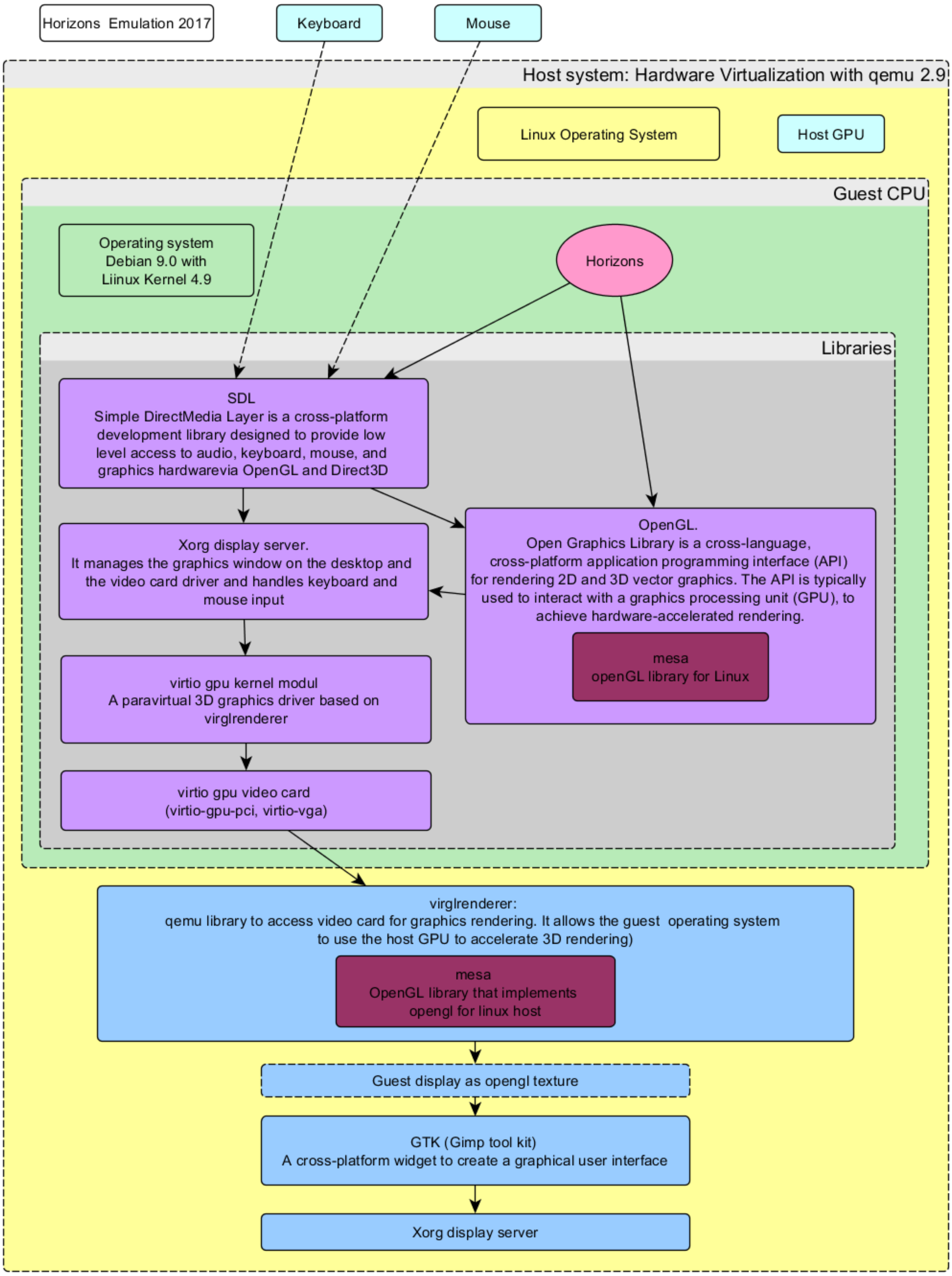


Figure 19: Virtualisation of *Horizons* by migrating it from Debian 7 to Debian 9 with QEMU

An overview of options for the handling of GPUs in emulation / virtualisation environments is given in Figure 20.

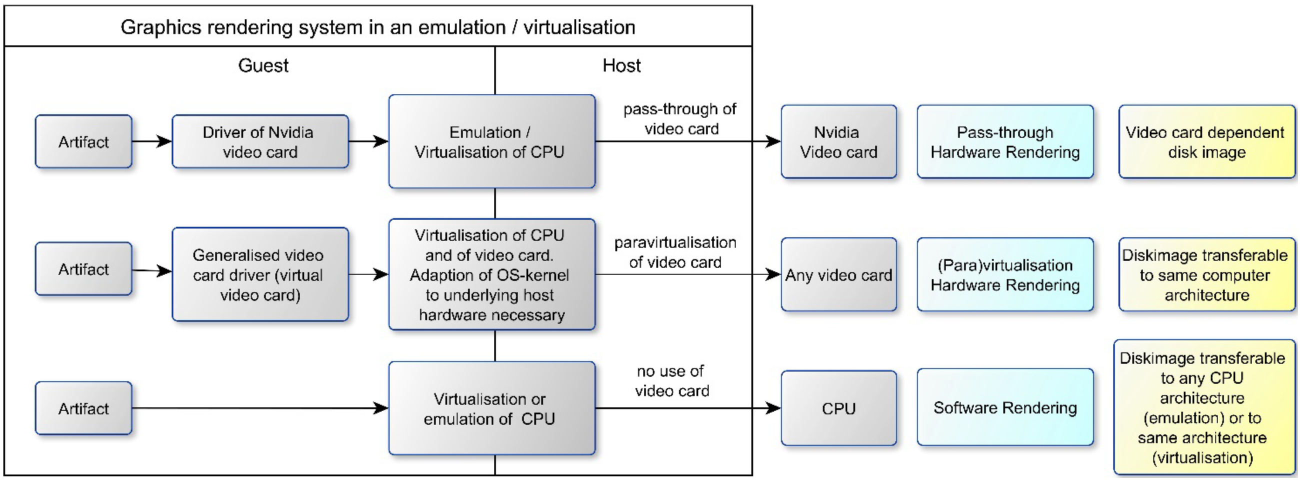


Figure 20: Emulation / virtualisation options for the graphics rendering of *Horizons*

Emulation / virtualisation options for the tracking sensor. The connection of the SICK sensor through a USB connector to the emulating laptop did not raise any concerns. The USB-input of the sensor was passed through to the guest system in the virtual machine without problems. With regard to the sensor usage, emulation / virtualisation is a valid option.

However, this does not solve the problem of the SICK sensor being replaced by a different sensor type. The SICK sensor is interwoven with the software artifact that is able to read the specific SICK sensor protocol (a manually written driver). If the sensor is replaced by a sensor with a different protocol, which is likely in the future, the software will not be able to parse it. Ideally - for preservation purposes - changes of the artifact or software setup should be avoided when replacing the sensor. This could be achieved with an additional software interface that translates the data of different sensor technologies and brands to the current SICK data protocol. While such an approach requires additional effort, adaptations of the software for the sake of preservation could be clearly held apart. Figure 21 depicts these options for the integration of the tracking sensor in an emulated / virtualized system.

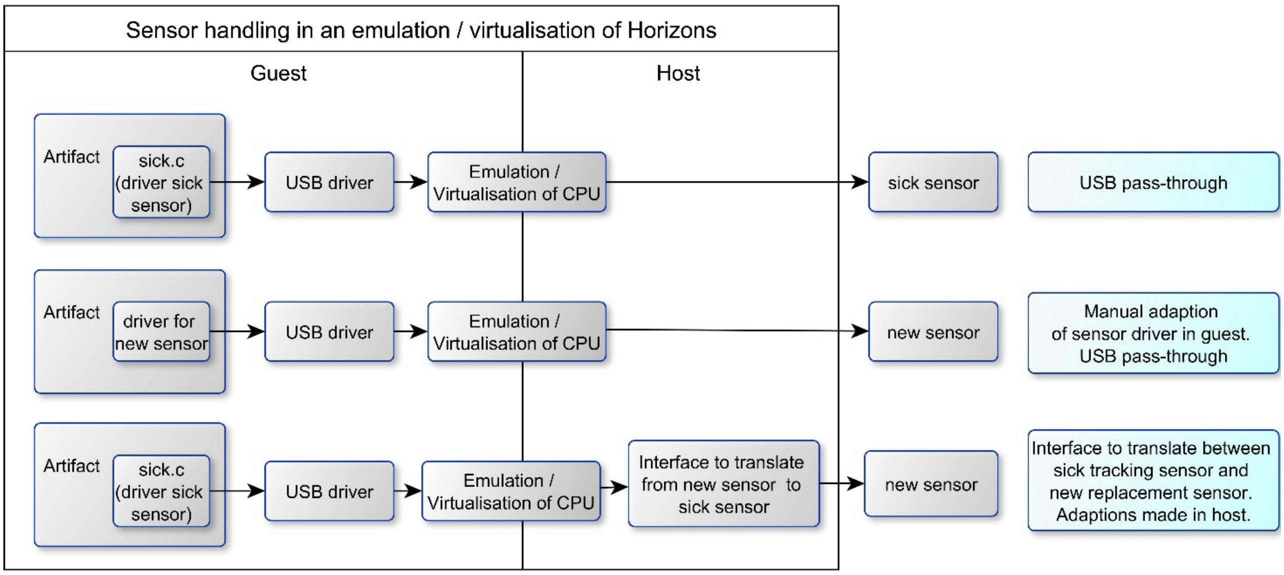


Figure 21: Emulation / virtualisation options for the tracking sensor of *Horizons*

4.7 Sensor and Screen Recordings as a Method to Document and Compare Elusive Significant Properties

Sensor Recording: Stabilizing the Input

Certain significant properties are difficult to quantify. For *Horizons*, the behaviour of the generated video is difficult to grasp due to the indefinite possibilities of visitor behaviour, due to the random processes built-in the software, and due to the complex interplay of the programs/processes. In addition, it is difficult to quantify the smooth rendering of the video, a property that is important to the artist. When comparing preservation versions or artwork instantiations, these elusive properties are important for their evaluation.

Thus, as a quality control for carried out preservation measures, we are suggesting to compare the video and sound output of different instantiations of *Horizons*. Yet, in order to receive comparable output, it would be necessary to carry out this comparison in the same space and based on the same person's walk across the space. This procedure is not completely reproducible. Consequently, the setup of *Horizons* was adapted, so that it became possible to record a person's walk, including the information about the space which the software considers as background. It also includes the specific program settings that were chosen in the "look-around" configuration menu of the custom-made software. All this information is saved in a file that can be replayed by the *Horizons* software instead of using

the sensor input. This stabilized input facilitates the comparison of different artwork instantiations.

The sensor input and therefore the file recording is sensor-model-specific. Different sensor models or technologies can therefore only be compared with each other in this way if an interface bridges the difference between the two sensor protocols as suggested in section 4.6 “Emulation / Virtualisation version 2017”. This interface provides the flexibility to make and use recordings with a new sensor while it is still possible to use the old SICK sensor recording. As long as the program that interprets the sensor input file does not change, any other software changes can be tested.

Screen and Sound Recording of Stabilized Input

The sensor recording enables the comparison of different screen and sound recordings, as it is now possible to produce them from the same sensor input.

From the tests carried out it can be concluded that the screen recording in conjunction with sensor input recording works very well in order to assess the generated video patterns. It is more precise than comparing documentary video recordings, which are made in various angles and with undefined visitor movements. Still, the smoothness of the movement cannot be checked with this procedure, as the process of screen recording reduces the graphics rendering performance. The impact of the screen recording on the rendering quality was noticeable, while running the artwork without screen recording yielded a smooth rendering.

For sound recording a special audio driver had to be installed in the guest system and the emulator's sound had to be redirected to that virtual driver. For the 2013 version the audio could only be captured through the analogue computer output, which resulted in a very poor recording quality.

Compliance with Significant Properties of the Artwork

The comparison of the videos and sound generated by different artwork instantiations was possible due to the recording of the sensor input. However, when comparing video and sound, one has to be aware, that the software randomly picks an image of varying resolution and a sound sample of varying length. This also has a certain, subordinate impact on the video and sound pattern and can be recognized by comparing several screen recordings of the same sensor input and the same hard- and software setup.

Compared to the 2013 version, the migrated version shows some deviations in the video pattern that are only noticeable in a direct comparison of screen-shots (s. Figure 22). For instance, at the moment of the snapshot taken in Figure 22, a fourth image is appearing in the migrated version whereas it is not visible in the 2013 version.

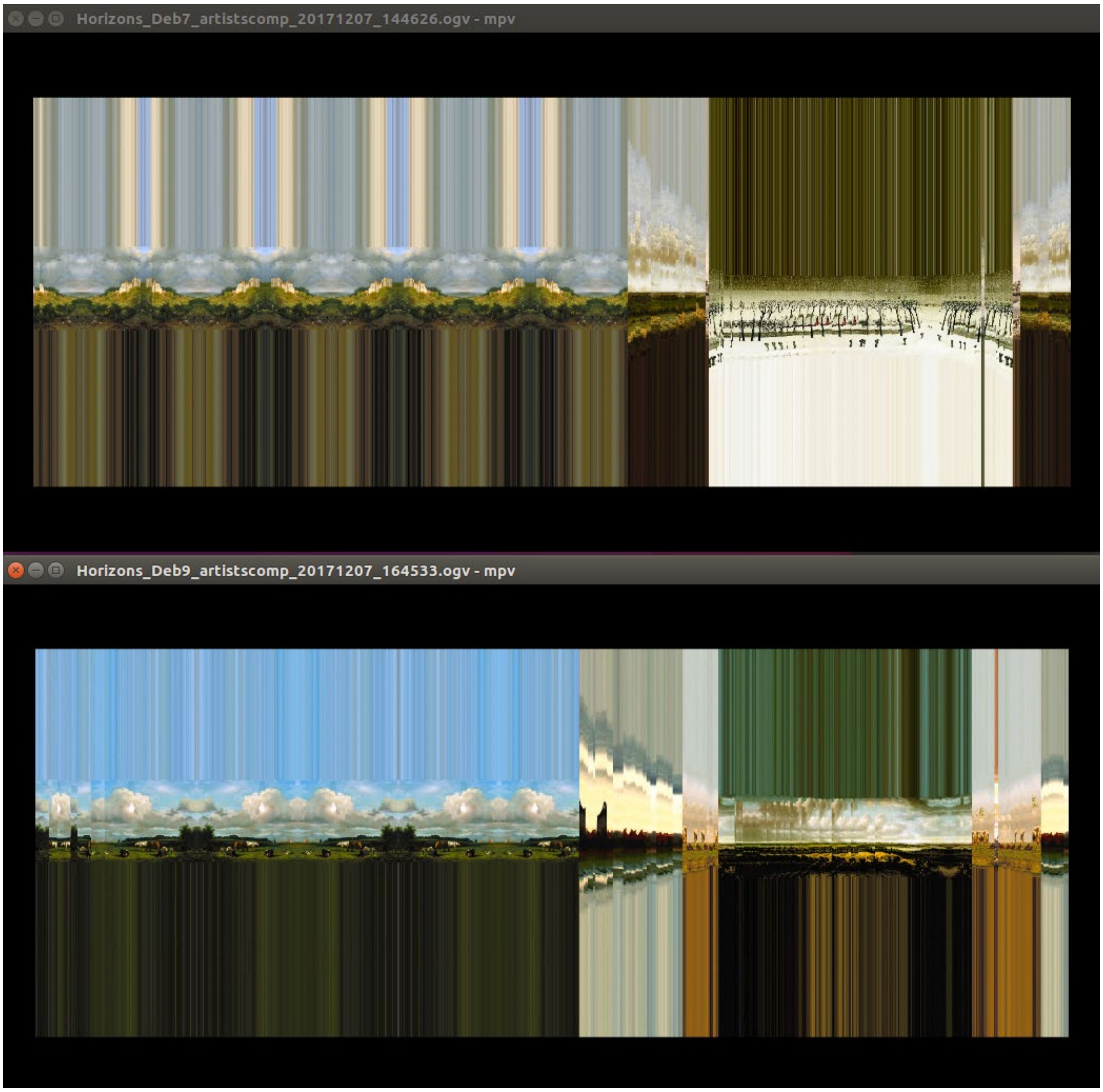


Figure 22: Corresponding pattern (of different paintings) in screen recordings made on the artist's computer. On the top *Horizons* version 2013, below the migrated version 2017.

The video patterns of the virtualised version are very similar to the ones of the migrated version. This was to be expected, as the virtualised version contains the migrated version in the guest system. However, the sound playback of the virtualised version was not satisfying, because it contained a disturbing crackling. This disturbance of the sound quality

would not be acceptable in an official representation of the artwork. As the sound recording of the virtualised version was of a good quality, it might be either a performance (sound dropouts) or a driver problem. Having tested several QEMU audio drivers and different emulated audio hardware, none of them yielded a satisfactory result. Other tests passing the sound through to an external USB audio card did not yield better results either. Hence, the work cannot be presented within an emulator at the moment. Sound performance issues are a known problem for QEMU, so is likely that it will be resolved in the future.

While the recording method served to compare the look and feel of the virtualised and the migrated version, the compliance with other significant properties resulting from the idea of the work and from the processes implemented by the software had to be checked, too. With both migration and virtualisation, the interactive nature of the work is preserved, meaning, that both react to a sensor input. Furthermore, both are directly executing the original code. This does not only comply with the significant property of code execution, but is also in line with article 8 of the E.C.C.O. directive which states, that preventive conservation should prevail over "physical" work and that treatment should be limited to what is necessary (E.C.C.O. European Confederation of Conservator-Restorers Organisations 2003). Except for the sound rendering in the virtualisation, it can be summarized, that both the virtualised and the migrated version sufficiently complied with the significant properties.

4.8 Discussion of Preservation Strategies Carried out for *Horizons*

The preservation strategies applied to our use-case deal differently with hardware and software dependencies. The main differences between an emulation / virtualisation and migration strategy, however, are found in future, reoccurring, preservation tasks. The migration strategy runs the risk of having to adapt the source code or replace the programming language in the future, as programming languages evolve or become obsolete. In a similar way the graphic system used (SDL/OpenGL/X11) eventually requires substitutes. These tasks are labour-intensive, difficult to automate, and their complexity is difficult to predict.

It has to be assumed, that all these updates and changes could have a perceivable accumulated effect on the generation of the *Horizons* video and on the complexity of the source code, as already the jump from Debian 7 to Debian 9 was visible (but not audible due to the noise quality of the sound). Hence, the complexity of the software artifact definitely

increases with the migration strategy in contrast to the emulation / virtualisation strategy that requires fewer changes and slows down the growth of complexity. However, the emulator itself might become obsolete and require a substitute.

At the moment, the virtualised version (in this case a paravirtualisation[33]) does not provide many advantages over the migrated version. It is rather a proof of concept. Despite of the current performance restriction, the virtualisation provides a better starting point for future, reoccurring, preservation work as it provides a higher level of abstraction. With the rapid improvement of hardware performance, there is a good chance that a pure emulation will soon be possible and adaptations in the guest system will no longer be necessary or less invasive. The most important advantage of the emulated version is that preservation work (e.g., its transfer to new hardware) is usually possible without specific knowledge of the artwork. In contrast, a migration does require more detailed knowledge about the identity and function of the software artifact and of its software environment.

For the installation of *Horizons,* the connection of peripherals poses the biggest risk. In the long-term, both preservation strategies will have to cope with new sensors and projectors. The installation of the projectors with the correct aspect ratio and resolution has not been tested with either of the preservation versions and might take some fine-tuning, especially for the virtualised version, as the video signal will have to be split onto two to three projectors. The connection of the SICK sensor worked for both versions. The connection of speakers is not expected to cause any problems for the migrated version. For the virtualised version, the sound playback will have to be resolved.

The following section expands on the preservation and preparation of *Shan Shui* (2013) for a permanent display at the Dortmunder U, three years later, in 2020. As mentioned earlier, *Shan Shui* is based on the same software and almost the same hardware as *Horizons*. Due to the sound problems of the virtualisation mentioned above, a migration approach was taken. The migration strategy carried out for *Shan Shui* is different, though, from the migration strategy applied to *Horizons,* as a package management system was used to deploy and document the migration.

[33] For a definition of paravirtualisation, see glossary.

4.9 Conservation of *Shan Shui* for the Dortmunder U

Mul created *Shan Shui* using the same custom-made software as *Horizons*. Instead of presenting the Museum Boijmans Van Beuningen's painting collection, however, *Shan Shui* presents pictures of Chinese landscapes as an interactive video, based on a database of about 400 images. As the visitor moves around in front of the projection, the video reacts, causing the image to split apart where they are standing and a new image to appear. As in *Horizons*, the sound, a kind of pink noise, also reacts to visitors' movements. In 2013, *Shan Shui* was set up with two adjacent 4:3 video projections and two speakers (see Figure 23).

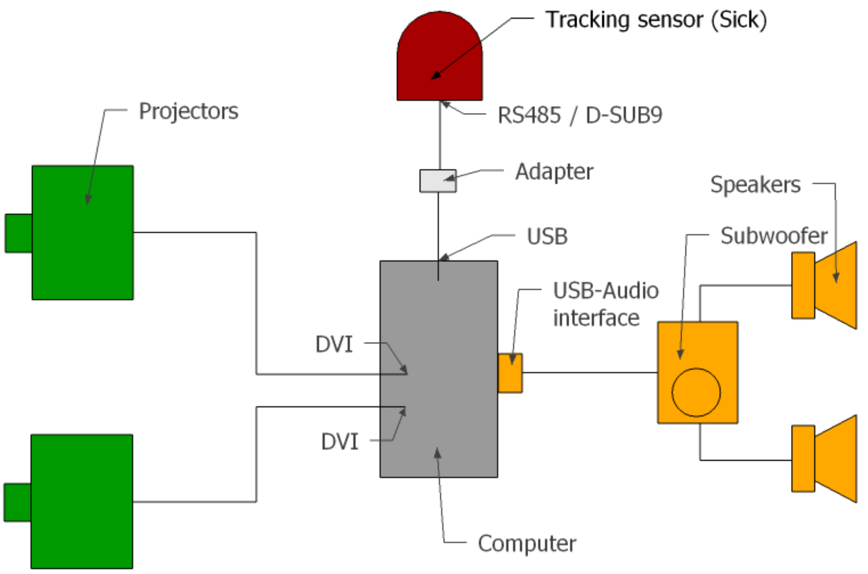


Figure 23: Equipment schematic for *Shan Shui* (2013) by Geert Mul

The primary differences between *Shan Shui* and *Horizons* are their image databases, the fact that the images in *Shan Shui* are not aligned according to their horizon (as is the case for *Horizons*), and the aspect ratio (8:3 for *Shan Shui*, 12:3 for *Horizons*), resulting in two adjacent projectors as opposed to *Horizons*' three. Otherwise, the software and hardware used in each piece are the same. The immersive experience is similar, and the behaviour of the video patterns is the same in both artworks.

In 2016, the Stedelijk Museum in Schiedam held a retrospective dedicated to Mul, where *Shan Shui* was installed alongside many of his other pieces (LI-MA 2016-2017a). The exhibition made possible the artwork documentation that informs the present case study, and contributed to my article for ipres2018, reproduced in sections 4.1 to 4.8.

In 2021, the Dortmunder U, a gallery in Dortmund (Germany), commissioned *Shan Shui* for permanent exhibition next to the gallery entrance. To facilitate the maintenance of

the work and adapt it to the space, Mul made certain decisions that he would not have considered for *Horizons*, as *Horizons* is already part of a Museum collection whereas *Shan Shui* belonged to the artist. He adapted the video's resolution and aspect ratio by changing parameters in the code and reconverting the images in the database to a height of 1200 pixels (from 768 pixels). This was due to the fact that Mul chose to use a single projector with an aspect ratio of 16:9, instead of two adjacent projectors with an obsolete aspect ratio of 4:3 each, resulting in an 8:3 aspect ratio overall. This worked better in the space and meant that any complexities involved in having to align two projectors were avoided. The gallery is not allowed to change the aspect ratio or resolution of *Shan Shui* without Mul's permission (see documentation in Gitlab [Roeck and Gaure 2021b]). Finally, the artist had to replace the tracking sensor with a new model based on a different data protocol as the old one had reached the end of its life. All these changes required software adaptations or bridges.

Due to the sound emulation problems encountered with *Horizons* (see section 4.7), we – the artist, the programmer Teal Gaure, and me – decided to carry out a strategy of migration using a software packaging system to deploy and migrate *Shan Shui*. The packaging system of our choice, Nix, is integrated with a Linux operating system called NixOS.

Not only did this allow the software to be rebuilt with the Nix package manager, it also means that the entire runtime environment – including Linux kernel and runtime configurations – can be reproduced.

In turn, this reproducibility means that changes such as operating system updates can easily be reversed.

Each Nix software package describes the relevant hardware and software configuration, and refers to each piece of software with a unique hash. As the hashed software is downloaded from an external repository (Nix package repository, [Nixos n.d.a]), hardware updates and transfers can be organised remotely.

One advantage of this approach is that it allows for accurate descriptions of the hardware and software configurations within the packaging system (see Figure 24).

flake.nix 602 bytes Open in Web IDE

```
{
  inputs = {
    nixpkgs.url = "github:nixos/nixpkgs/nixos-20.09";
    boijpkgs.url = "gitlab:shanshui/boij";
  };

  outputs = { self, nixpkgs, boijpkgs }: {
    nixosConfigurations.shan-shui-dortmund = nixpkgs.lib.nixosSystem {
      system = "x86_64-linux";
      modules = [
        ./shan-shui/dortmund.nix
        { nixpkgs.overlays = [ boijpkgs.overlay ]; }
      ];
    };
    nixosConfigurations.shan-shui-usb = nixpkgs.lib.nixosSystem {
      system = "x86_64-linux";
      modules = [
        ./shan-shui/usb.nix
        { nixpkgs.overlays = [ boijpkgs.overlay ]; }
      ];
    };
  };
}

```

Figure 24: *Shan Shui* as set up for Dortmunder U in 2021: hardware and operating system configuration (Gaure 2021)

However, there are downsides to this approach:

a) The accessibility of the external software repository, whose continuity depends on an organisation having priorities other than the preservation of digital heritage, is a risk that needs to be mitigated, for example by downloading the required software.

b) As the computer hardware will have to be replaced periodically, it will also be necessary to regularly update the software environment. At some point this may affect the source code of *Shan Shui*, as this may also have to be updated to newer Ruby or C versions. Thus, the work is changing slowly, and migrations will have to be executed repeatedly.

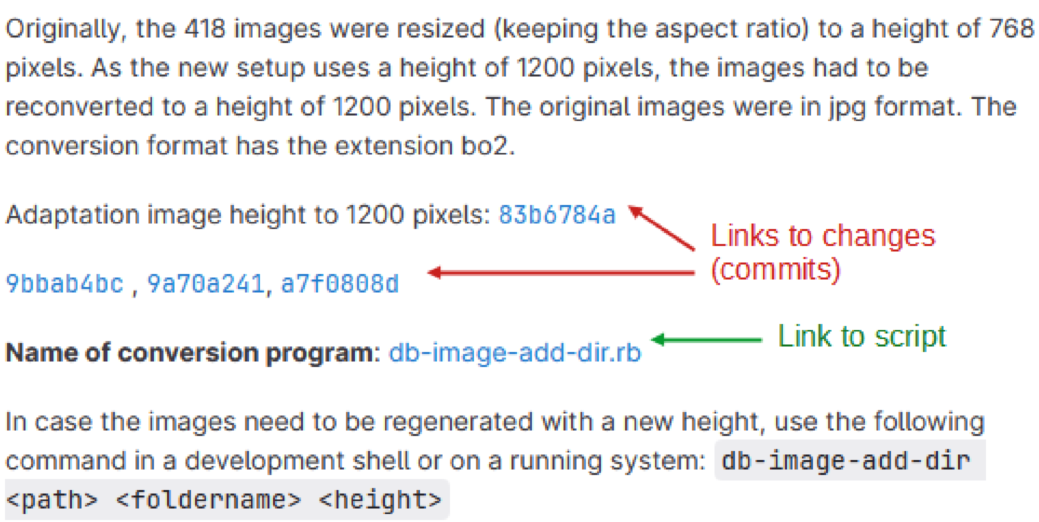
c) Although NixOS is a Linux operating system, the software environment does not conform to the Linux Filesystem Hierarchy Standard (Linux Foundation n.d.). Software is installed in the Nix/store directory and the name of the

software is changed to hashes. Proficiency with the Nix language[34] is necessary to be able to handle the NixOS environment.

d) To install Ruby files on NixOS, it was necessary to make a Ruby package. For this, the Ruby and C files had to be reorganised in different directories,[35] and Ruby had to be updated from version 2.1 to 2.7.

Whilst this migration approach does gradually change the artwork, it is suitable for cases where emulation is not possible, and allows proper management of the migrations in question (reversibility, reproducibility, runtime description).

Independent from the packaging system approach, we managed the software and the packaging scripts within a version control system, which facilitates documentation of the software and any changes made to it. The version control system that we used comes with an integrated Wiki, which allowed us to link and describe source code changes (commits) and conservation decisions (see example in Figure 25). Furthermore, this setup allows the artist or museum to fork[36] the project and adapt it without changing the original code.


Originally, the 418 images were resized (keeping the aspect ratio) to a height of 768 pixels. As the new setup uses a height of 1200 pixels, the images had to be reconverted to a height of 1200 pixels. The original images were in jpg format. The conversion format has the extension bo2.

Adaptation image height to 1200 pixels: 83b6784a

9bbab4bc , 9a70a241, a7f0808d

Name of conversion program: db-image-add-dir.rb

In case the images need to be regenerated with a new height, use the following command in a development shell or on a running system: db-image-add-dir <path> <foldername> <height>

Figure 25: *Shan Shui* as set up for Dortmunder U in 2021. Wiki documentation of source code changes (Roeck and Gaure 2021b). This example documents the adaptation of the image height mentioned earlier.

34 Nix language reference manual see (Nixos n.d.b).
35 Details see (Roeck and Gaure 2021c).
36 Forking is a term used in source code version control systems. It means copying the whole software project (source code) with all its branches and changes including documentation.

4.10 Conclusions for *Horizons* (2008) and *Shan Shui* (2013)

The behaviour of interactive, software-based artworks such as *Horizons* or *Shan Shui* is far too complex to describe with sufficient precision as to reproduce their software after the fact. Moreover, as the source code consists of several thousand lines, translating into pseudo-code and from there into another programming language would be expensive, and the result would most likely deviate from the use of the original source code. These practical factors lead to the definition of source code as a significant property of *Horizons* and *Shan Shui*. As Pip Laurenson describes (Laurenson 2014, 92–94), significant properties can also include dependencies. Indeed, source code can be considered not only as a detailed notation of processes and algorithms, but as a dependency, as it is difficult to replace in practice.

No clear winner emerged from section 4.8's comparison of virtualisation and migration mainly because the paravirtualisation[37] conducted there relies on virtualised hardware (a video card), and remains dependent on computer architecture for performance reasons. Moreover, the emulation of sound was not satisfactory, a problem for future emulators to solve. However, if virtualisation is seen as the transition to full-system emulation in a few years when computer performance will have increased, paravirtualisation is promising.

The most significant advantage of the virtualised version – essentially a result of combining migration and virtualisation – is that transferring it to new hardware is possible without specific knowledge of the artwork. Migration, by contrast, does require such knowledge. Furthermore, virtualisation reduces the change rate of the software artefact. The combination, or merging, of virtualisation and migration is therefore a strategy that may prove successful in the mid-term, provided the new hardware does not have enough of a speed advantage to enable a full-system emulation. Furthermore, the virtualisation of the migrated version allows for the effects of virtualisation and any hardware dependencies to be identified, and serves as a sandbox[38] for migration.

With the emulator or virtual machine, the artwork is encapsulated; its behaviour, therefore, does not need to be understood and described in minute detail. To still be able to compare preservation versions, any potentially elusive significant properties can be partially captured by recording video and sound from a stabilised input.

[37] For a definition of paravirtualisation, see glossary.
[38] In software engineering, a sandbox is a testing environment.

The encapsulation of an artwork has another benefit: it protects and separates the work from changes introduced later. The sensor interface that could translate other sensors to the current SICK sensor protocol would support the replacement of the tracking sensor. This interface and its connection to the emulator is something that future research could explore.

Using the package manager Nix to migrate *Shan Shui* made this migration reproducible, and automatically created a precise description of the software and its dependencies. Installing the artwork software on a special Linux operating system (NixOS), integrated with the package manager, made it possible to not only reproduce the software installation, but the entire runtime environment, including the Linux kernel and runtime configurations. In contrast to emulation, the runtime remains hardware-dependent, but a precise documentation of all the dependencies is guaranteed. However, were the software in need of being reinstalled with the packaging manager, its dependency on an external software repository for generic components is a big disadvantage in terms of sustainability.

This chapter has shown that paravirtualisation can be a good strategy for interactive, graphics-intensive works in the mid-term. It steps in between short-term solutions such as migration, and long-term solutions such as a full-system emulation. Furthermore, I have also explored how a package manager can be used to reproduce migration and create automatic descriptions of both an artwork's runtime and dependencies.

5 Case Study 2: *TraceNoizer* (2001-2004) by LAN. Preservation Strategies for an Internet-Based Artwork Yesterday, Today, and Tomorrow[39]

This chapter presents my second case study, the internet-based artwork *TraceNoizer* (2001-2004) by the artist group LAN. Whilst this software-based work is less hardware-dependent than contained artworks, it does depend on a networked environment, the World Wide Web. Since 2001, the World Wide Web has undergone enormous changes, and these have had a major impact on the functioning of the artwork.

I investigate possible preservation strategies for *TraceNoizer* and assess those that best capture the work's authenticity for future iterations. I discuss my experience applying two different strategies, as well as another, a Linux Live CD containing *TraceNoizer*, that one of the members of LAN carried out in 2004. I then compare these three strategies and evaluate them, in terms of fidelity to the original artwork and sustainability, in sections 5.1 to 5.7. Besides preservation strategies, I also examine *TraceNoizer*'s display history in section 5.9, and the relationship between these different presentations and the preservation of the piece.

Compared to contained software-based artworks such as Geert Mul's *Horizons*, internet-based artworks shift the focus of preservation measures away from stabilising software to reducing server maintenance, protecting both the server and artwork from internet threats, and reducing the number of external dependencies. This chapter proposes ways to approach these challenges and discusses their benefits and disadvantages for long-term preservation.

5.1 Introduction

Until a few years ago, Internet-based art was not widely collected in contemporary art museums and collecting institutions. With the acquisition of *net.flag* (2002 by Mark Napier) in 2002, the Guggenheim Museum in New York was one of the first contemporary

[39] Sections 5.1 to 5.8 and section 5.10 of this chapter are based on
Roeck, Claudia, Klaus Rechert, Julia Noordegraaf, and Rafael Gieschke. 2019. "PRESERVATION STRATEGIES for an INTERNET-BASED ARTWORK YESTERDAY, TODAY and TOMORROW." In *IPres2019: Conference Proceedings*, edited by iPres2019, 179–90. Amsterdam. https://ipres2019.org/static/pdf/iPres2019_paper_125.pdf and https://hdl.handle.net/11245.1/3b90f2b6-d6c0-4fc0-adc9-63c166f23307.

art museums to acquire web-based art. Rhizome, an art organization in New York, has probably the widest experience in the preservation of Internet art. In contrast to most museums, they are focusing on purely digital art. Their collection consists of several hundred digital artworks. The number of museums collecting internet-based art is slowly increasing. For instance, the Stedelijk Museum Amsterdam (NL) acquired several internet-based artworks jointly with the MOTI museum in Breda (NL) in 2017, LI-MA in Amsterdam is hosting internet-based artworks by and for Constant Dullaart since 2018 and the House of Electronic Arts in Basel (CH) acquired about 20 internet-based artworks between 2016 and 2018. While the museums slowly start to acquire Internet art, the preservation, change management and hosting of these artworks is often not solved. This is the reason why an internet-based artwork was chosen as a case study for this research. Its embedding in the Internet is a specific feature relevant for its preservation that does not exist for contained software-based artworks such as *Horizons* (2008). It makes the work vulnerable towards changes of the internet environment. This article shows different ways for dealing with the long-term preservation challenges of an internet-based work. They are demonstrated on a case study, *TraceNoizer* (2001-2004) by LAN, acquired by the House of Electronic Arts (HEK) in Basel in 2017. After defining its significant properties, different digital preservation strategies are applied, their sustainability and their impact on the authenticity of the artwork examined and compared. One of the artists carried out a preservation strategy already in 2004, which offers an interesting opportunity to study its long-term effects. Finally, the differences between the preservation strategies for contained software-based and internet-based artworks will be discussed.

5.2 Related Works and Definitions

Annet Dekker describes the performative nature of internet-based works in her dissertation (Dekker 2014, 162). In her view, because of the great variability in their appearance and behaviour, it is not possible to conserve an actual net artwork with traditional approaches (Dekker 2014, 55). While we also find that it is not possible to reconstruct the actual, original appearance and functionality of *TraceNoizer*, we do propose strategies for preserving the significant properties of this work for future iterations. Phillips et al. report the restoration of Shu Lea Cheang's early web-based artwork, *Brandon* (1998-1999), at the Guggenheim Museum in (Phillips et al. 2017). They migrated the work (about 65'000 lines of code) to a current web server and annotated the changes they had to carry

out in order to make it compatible with today's web browsers (Engel and Phillips 2018, 4, and Engel et al. 2018, 31). This huge effort was done with the help of computer science students. Future migrations will necessitate repeated efforts and introduce changes with each migration. Not many institutions have the means to do repeated migrations every few years. Hence, this paper will compare different preservation strategies in order to find more sustainable solutions.

Miksa, Mayer, and Rauber are proposing strategies for businesses whose processes depend on web services (Miksa, Mayer, and Rauber 2015, 78). They are suggesting creating "mock-ups" for these web services. These mock-ups do not actually process requests but instead pull the response from a database of the mock-up. For this purpose, they recorded the request and response streams of this web service. Espenschied and Rechert suggested this "mock-up" strategy for "Apparently Infinite Objects", in particular internet-based artworks (Espenschied and Rechert 2018, 2). Implementation, its feasibility, and efficiency, remain open. Besides, Espenschied and Rechert proposed a stub interface and the mirroring of web services to deal with external dependencies, that will be discussed in section 5.7 and 5.8 of this paper.

Web archiving can be applied to preserve certain or parts of internet-based artworks. In 2014, Mat Kelly highlighted in a presentation (Kelly 2014, 22), that a web crawler usually changes the capture context of the web browser. As a consequence, he formulates high level requirements for the creation of web archive files (WARC). In particular, he asked for a crawling software to capture the embedded scripts of a web page and to allow the user to execute these WARC files in a web browser. Rhizome subsequently developed such a system with the Webrecorder and Webrecorder-player (see interview with I. Kreymer, 2016, and github.com/webrecorder). These approaches, however, are only capable of capturing the "surface", for a truly internet-based artwork and thus have to be extended with solutions that also preserve its logic. For this research, we are building upon Roeck, Rechert, and Noordegraaf (Roeck, Rechert, and Noordegraaf 2018) using the same method (determine significant properties, compare different strategies and evaluate based on criteria for long-term preservation). However, the analysis was not geared towards internet-based artworks which is why they deserve further attention.

5.3 Significant Properties and Dependencies of *TraceNoizer (2001-2004)*

Starting From a Digital Ruin

TraceNoizer is an interactive website created by the art collective LAN (Local Area Network: Annina Rüst, Fabian Thommen, Roman Abt, Silvan Zurbruegg and Marc Lee) in 2001. Anybody who had access to a computer with a screen, keyboard, mouse and Internet connection could experience it. The House of Electronic Arts in Basel acquired the work in 2017 as a pile of code. *TraceNoizer* had not been online for many years. Even though the source code was available, it was not fully functional.

As it is common for institutions to acquire works a few years after their creation, many websites such as *TraceNoizer* have to be reconstructed without having a functional reference. The project archive delivered by the artist is by itself a blurry object, insofar as its work-defining properties are not completely known and the code that is relevant for the work not clearly delimited. The plethora of versions and customized presentations, adaptations to free web services and APIs, improvements of functionality and web design, multiple backups, handwritten maintenance programs and the simultaneous lack of documentation turns the archive into a maze for a conservator or curator. Even though the artwork itself might not have been blurry, its data, which is the base of reconstruction, might be. Hence, the process of the definition of significant properties is split up in two steps:

1) The work will be described as a reconstruction of the past (what we think it worked and looked like in 2004. The 2004 version is the most recent version. It is based on the same code as the Conservix CD, s. Figure 26)

2) Based on that, the significant properties of the artwork restoration will be determined (transfer to the present and future).

As this process and its results are subjective, the decisions will be supported by reasons so that others can understand the motivation to determine certain properties as significant. Hence, *significant for restoration* means *considered significant for restoration*.

Figure 26: *TraceNoizer* by LAN. Screenshot Conservix CD (Knoppix CD)

Idea of the Work

In the beginning of the 2000s, the aspect of data autonomy was widely discussed among critical Internet users. The fact that it was often impossible to delete one's own traces in the Internet motivated LAN to create *TraceNoizer*. The work was inspired by the "Jam Echelon Day" on the 21 of October 1999, when the international hacker scene decided to flood surveillance agencies such as CIA with fake information. LAN applied a similar counter strategy with *TraceNoizer* in order to diffuse the traces of one's own personal data on the Internet. On the *TraceNoizer* website the user could enter his/her name. Subsequently, the clone engine searched for all the websites containing this name and reassembled a new personal website out of these search results by using an algorithm that follows a similar logic as the ranking of the search machines. Thus, theoretically, in a following search, the search engine should rank the generated website higher than the original ones. The generated websites (so called clones) were uploaded to free website hosting platforms that indexed these new clones so that there was a chance for the search

machine to find them. The more this process was repeated, the higher up climbed the ranking of the clones until the “original” pages did not appear in the search results anymore.

The artists highlighted in an interview (interview with F. Thommen, M. Lee and A. Ruest, 2017, June 22, 24), that *TraceNoizer* was a performative tool and not a static website. According to the same interview, another important aspect of the work was the automatic generation of websites without human interaction except for entering their name.

The artists called the generated websites ‘clones’, which is why this article continues to use the term clone, although they are not clones in a literal sense, but assemblies of text and images from other websites. The programs that produce the clone are subsumed under the term *clone engine*.

Significant for the restoration: The cloning process was supposed to be repeated until the clones themselves appeared in the search results and were used to produce the new clones. However, there were some discussions among the artists, whether the high search engine ranking of the generated clones really worked. This doubt is confirmed by jury members of the readme festival (2002) where *TraceNoizer* was exhibited. According to their experiences, their clones would not appear in the search results. Hence, the original idea was not perfectly executed in the 2004 manifestation of the work.

Processes Implemented in Software Version 2004

The *TraceNoizer* logic consists of three main parts: the graphical design of the website, the clone engine where the clones are generated and uploaded to the Internet and the clone control center where the user could access his/her clones (s. Figure 27). The graphical design will be discussed in the sections “Look and Feel” and “External Dependencies”.

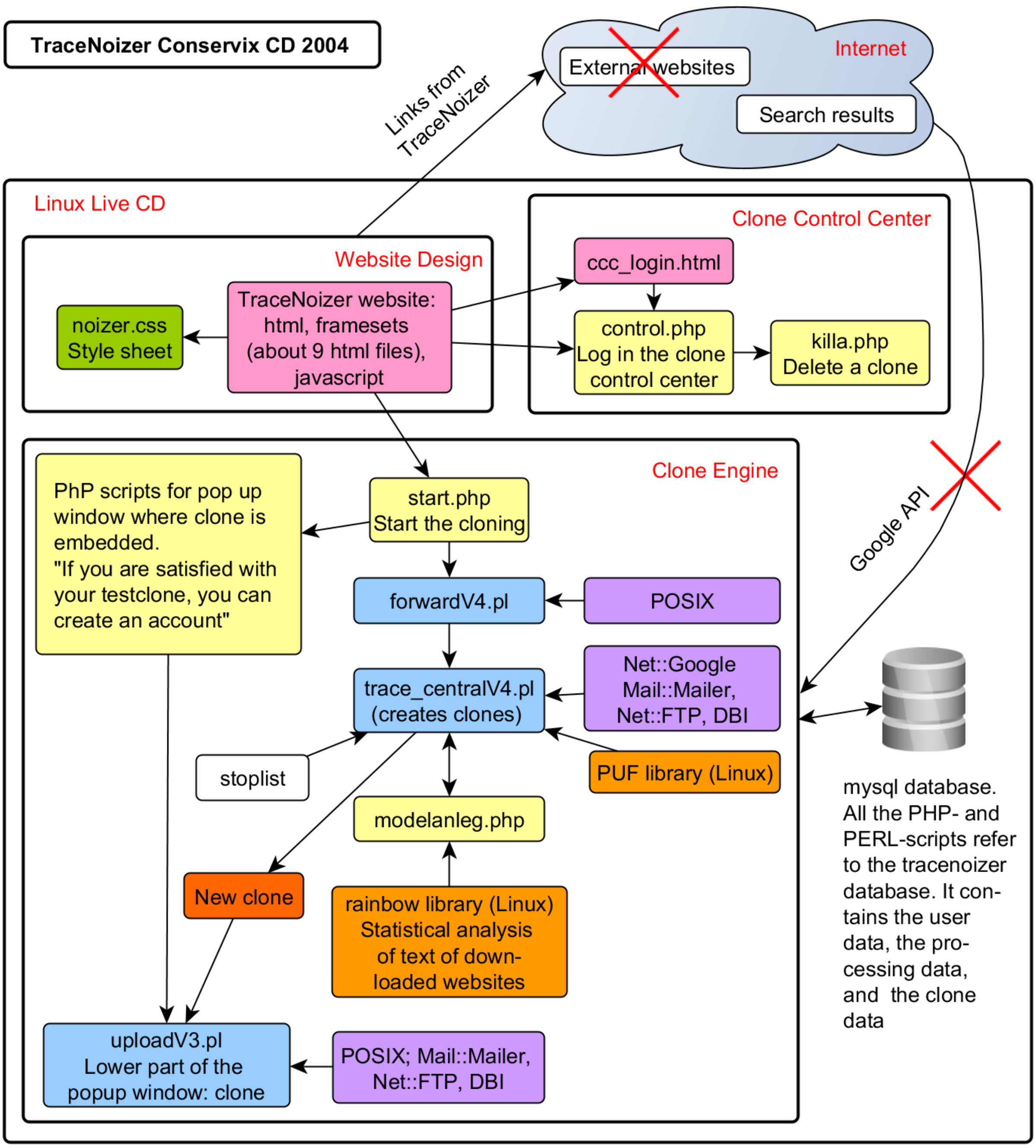


Figure 27: *TraceNoizer* in the Linux Live CD version (Knoppix CD)

The PERL script trace_centralV4.pl represents the most important script of the clone engine as it creates the clones. Another central element is the database that serves as a temporary data storage for the clone generation and a permanent storage for the user logins and the location of the uploaded clones. After the user entered the search term (the user's name, but other names can be entered as well), the clone engine started to search the Internet via a search engine interface. Up to 40 search results (websites) were saved in the database. Their text was extracted, temporarily saved in text files, then the text was split up in sentences that are saved in the database. The text was statistically analysed using the

rainbow library[40]. The ten most frequently used words were saved as keywords in the database. The texts and sentences were ranked according to the frequency of the keywords. This was the basis for the generation of the clones that mimic personal websites of the previously entered name. Each clone had a main page with ten sub-pages, each of these sub-pages corresponding to a keyword. Each keyword page consisted of an image, a text and an external link. The clone was then uploaded to a free host and the temporary database entries and text files were deleted. If the user was not satisfied with the created clone, he/she could log in the clone control centre and delete the clone from the host. The user login could be different from the name entered in the search. However, each user could only manage one clone.

Significant for the restoration: The scripts and programming languages are the material of the website. The way the scripts are programmed is typical for the period of the early 2000s and hence considered as significant for preservation.

Look and Feel: Graphical Website Design

The look and feel of the *TraceNoizer* website can be investigated by running the Conservix CD with Mozilla Firefox 1.3 released in 2002. There are no requirements for the web browser such as specific plugins. The monitor resolution common at this time was 1024 x 768 pixels.

The graphical design consists of a puzzle of website pieces (frames) that are held together by the index.html-page and are reused for the different sub-pages. This feature is not supported in HTML 5 anymore. The background of the links change colour, if the mouse hovers over it. This is achieved by JavaScript. While the website design is very clear, logical, and functional, from a present-day perspective the design looks a bit outdated.

Not only the *TraceNoizer* website needs to be evaluated, but also the generated clones. The clones are built much simpler (s. Figure 28). They just consist of a title, a centred image, and sub-pages. The sub-pages also have a title, a centred text and an external link on the bottom. Except for the images they do not contain any graphical elements. Even for 2001 standards the aesthetics of the clones are crude.

40 Information about the rainbow library: (McCallum 1994 / 2002).

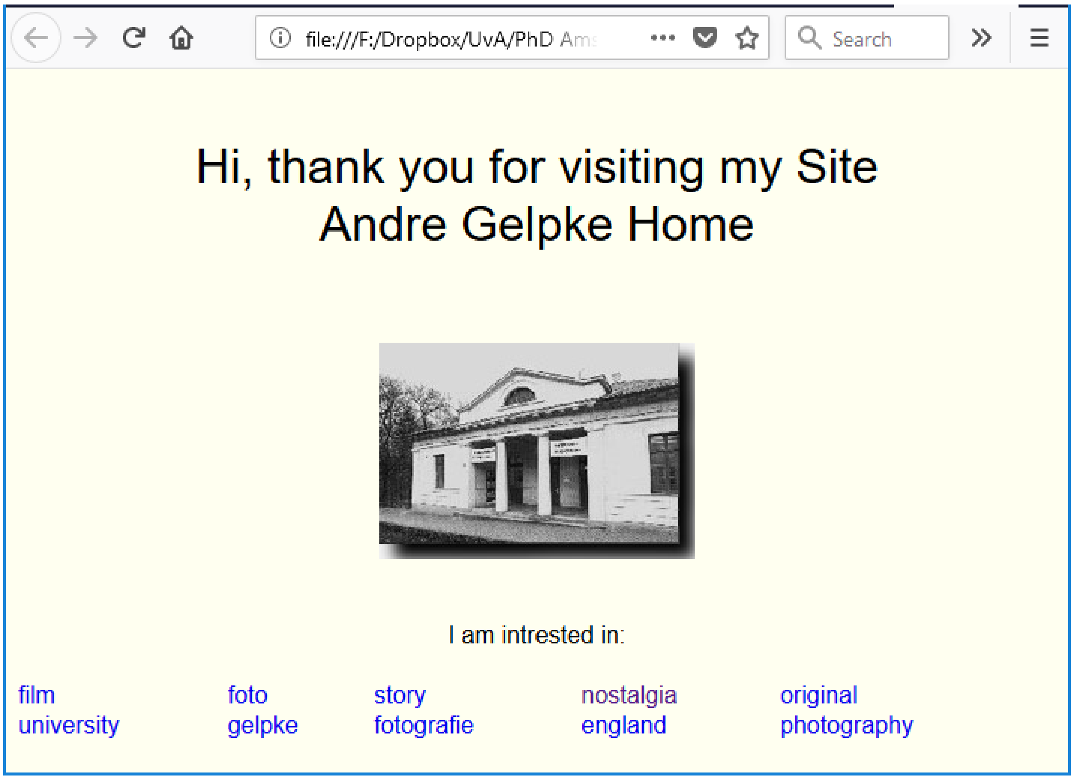


Figure 28: Clone from the clone-it project 2001 in Firefox 60 (2018)

Significant for the restoration: The look and feel of the *TraceNoizer* website is considered as a significant property as it points to the time of its creation. As a second priority, the framesets should be preserved, as a typical technology for that specific period.

As the clones had already looked quite crude in 2004 (They just consisted of text, images and links), there is no reason to change the clone design today. Hence, the clones generated with the clone engine today should look the same as they looked back then.

Environment

TraceNoizer contains external links that reach into the World Wide Web. Other artists such as Darko Fritz or Knowbotic used the clone engine to create their own artworks. The links to both artworks are broken. LAN themselves used the clone engine for two events in 2001 and produced a plethora of clones. These clones are not online anymore, but the artists have stored them on the Conservix CD.

Significant for the restoration: The above-mentioned web pages of related artwork projects are considered significant, as *TraceNoizer* links to them directly and the artists used the clone engine to create these projects. Hence, these web pages should be preserved as context of *TraceNoizer*. It could even be argued that the indirect environment such as the artist's websites, web pages from *TraceNoizer* exhibitions and the Echelon-web page should

be preserved as context, which is a second priority. This context demonstrates the performativity and project character of *TraceNoizer*.

External Dependencies

Dependencies that reach outside the artwork's web server to web servers that the artist and the collecting institution have no control over are typical for internet- based artworks. For *TraceNoizer*, this applies to the use of the Google search engine. The PERL interface to the Google search engine used for this work has been obsolete since many years and does not work anymore. The upload of the clones to free hosting platforms is another fragile dependency, as the free hosting platforms and their requirements to host pages for free change often. The artists experienced that too and asked friends for free server space to store the clones. For this reason, the FTP clone upload was deactivated in the 2004 version of the code. Finally, *TraceNoizer* depends on the HTML code of websites of the period around 2004. The clone engine extracts text, images and links from websites in order to reassemble them later. If this extraction does not work well, the newly assembled clones will be faulty. However, there must have been a certain percentage of faulty clones already in 2004 due to the automatic clone generation.

Significant for the restoration: The different versions of the clone engine show that the artists switched from Google search engine in 2001 to Yahoo in 2002 and back to Google in 2004. The Google search engine itself changed considerably since 2004[41]. How the changes affect the artwork is not clear, but the artists did not seem concerned about it. In order to be consistent with the creation period a search engine from 2004 would be preferable to a current one. Knowing that this is not possible, any search engine based on the page rank algorithm (Khosrow-Pour 2013, 508) is acceptable.

Due to the fact, that LAN stopped uploading the clones to free hosting platforms, the accessibility of the clones on the Internet, but not the use of free hosting services is declared as significant.

The legibility of the clones is significant for the understanding of the work. Hence, most produced clones should be legible. In other words, they should contain legible text (not html code) and images.

---

[41] "With some 1600 improvements to Google Search in 2016 alone, these are just a sample of some of the ways we have been making Search better and better over time." (Google 2019).

Conclusions

Due to nature of internet-based artworks, dependencies on external services and external websites that are part of the ever-evolving network are a real challenge. These external dependencies can be split up in technical dependencies and content dependencies. Espenschied's technical definition of a blurry object in (Espenschied and Rechert 2018, 1) refers to technical external dependencies. Such external technical dependencies as the Google search engine in *TraceNoizer* cause variations in their execution, as these dependencies change. In contained software-based artworks, similar variations are caused by replacement of hardware such as computer and peripherals. The blurriness of the object extends beyond its technical external dependencies, in that internet-based artworks also have content-related dependencies with external objects/sources. As it is usually not possible to confine a context of a net-based artwork, its delineation is often subjective or dependent on circumstances. Strictly speaking one could argue that the clone-it project and the links to external websites or related art projects is not part of the work, but in order to fully understand the purpose and effect of *TraceNoizer* they become part of the work. This kind of interlocking with the environment is typical for internet-based artworks. The performativity of *TraceNoizer* consists of the clone generation (clone engine) and the clone management (clone control center) by the user. The fact that the input is generated by search results from the Internet poses challenges for preservation, as the properties of the Internet are gradually changing. Finally, the user management and databases are a typical property of networks. New user specific data has to be saved and becomes part of the work. Although user specific logins might also be part of certain off-line artworks, it is much less common, and it is limited to the visitors of the physical artwork.

5.4 Long-Term Preservation Criteria

In order to find and evaluate preservation strategies for the above-mentioned properties typical for internet-based artworks, the following criteria for sustainable preservation strategies are suggested. Criteria for assessing the long-term sustainability of preservation measures were established for contained software-based art in (Roeck, Rechert, and Noordegraaf 2018, 4–5) and are hereby assessed for web-based artworks.

Adaptability to new hardware is relevant but not crucial for the backend. Web servers are generic computer hardware without specific components such as super-fast video cards or specific input or output devices. Almost the same can be said about client

computers. In order to view the artworks, they do not use any specific hardware features except for certain generic input devices.

The *ability to deal with software obsolescence and changed network protocols* is very relevant, as the software of an internet-based work is usually updated when transferred to a newer web server. In addition, the work needs to adapt to new network protocols that are up-dated periodically (for instance HTML every few years). The change of application programming interfaces (API) is another frequent cause for malfunction internet-based artworks and an example for software obsolescence.

The stabilization of software complexity and the minimizing of the software change rate is very relevant for web-based artworks, as they are subject to fast and frequent changes due to the Internet connection. As their environment changes fast, the risk is high that such works quickly become outdated if not dysfunctional. In order to prevent that, they need to adapt, too.

The ease of installation of the artwork and of connecting peripherals is not relevant for web-based artworks.

The reduction of maintenance is very important for internet-based artworks, as maintenance of internet-based artworks, especially of server-side dynamic websites, can be laborious. Tasks such as the detection and cleaning of abusive and resource-intensive processes, updating the server software, and maintaining the database and server health can take up to one hour per week, which, in a museum context, is quite intense.

The scalability of a preservation strategy is relevant for web-based artworks, depending on the number of artworks an institution hosts. The more artworks, the more methodical the hosting and preservation approach needs to be in order to be able to exploit synergies and to reduce the maintenance per artwork.

5.5 Linux Live CD (Conservix CD)

In 2003, Fabian Thommen produced a CD-ROM with a bootable live operating system based on the knoppix-technology that he named Conservix. Conservix is set up with a basic Linux operating system, an Apache web server, a MySQL database, the programming languages PHP and PERL and the web browser Mozilla Firefox 1.3[42]. With this live system, the user did not need to install anything or change computer configurations.

[42] Operating system Debian 1.3-4, web server Apache 1.3-27, database MySQL version 11.18, distribution 3.23, programming languages PHP (4.1) and PERL 5.8, and web browser Mozilla Firefox 1.3.

The computer starts from the CD in place of its own boot system. It automatically opens *TraceNoizer* that is installed as a dynamic website (s. Figure 26).

The Conservix CD fulfils the previously defined *significant properties* partly. It was possible to temporarily create clones with it, although they could not be uploaded to external web hosting services, and the database entries could not be permanently stored as the CD is read only. For this reason, the user could not manage the clones in the clone control centre, partly diminishing the performative nature of the work. The client computer and the web server coincide in one machine. An important disadvantage of the CD is, that it cannot be directly accessed through an URL, but needs to be installed in an emulator in order to run. Due to the obsolescence of the search engine interface, the CD-ROM does not allow to generate clones today, while the website as a graphical interface is displayed without errors. The CD-ROM also comprised the clones of the clone-it project so that it is known today, what the output of the clone engine looked like. Unfortunately, the input stream (search results) of the clone engine was not recorded, so that it is not possible to verify today, whether a restored version would produce the same output. Other "damages" are external links on the *TraceNoizer* website that are broken in the meantime.

Regarding the *sustainability* of the Conservix CD from 2003 it can be stated that its iso-image still exists in 2019 and that an Intel or AMD-compatible processor can still run it. The Conservix CD is able to deal with software obsolescence and changed network protocols such as HTML as it contains both server and client. When it comes to external dependencies such as the Google API, Conservix does not cope so well. At least, it would be convenient if the CD produced an error message saying that the Google API does not work anymore. The Conservix CD stabilizes software complexity as it is read only. The maintenance is also low, but would include the periodical updating of external dependencies such as Google API and thus producing a new CD-ROM. The security risks are zero for the host computer, as the CD is read-only, and the computer does not use its own operating system. The preservation strategy is scalable in so far, as Linux Live CDs can be produced for other internet-based artworks.

The Conservix CD played an important role in the definition of the significant properties: It documents, how the work is installed, what libraries and program versions, and what browser were used. It recreates the look and feel of the work without having to install much, except for a generic emulator. On the other hand, a Linux Live CD is not the best solution for internet-based works with external dependencies such as a Google search API. It also gives a false sense of security, as Fabian Thommen mentioned that he did not install

the work exactly the same way as on the web server. For instance, administration programs to maintain the database are not necessary on a CD ROM.

5.6 Preservation Version "Migration"

This section describes the preservation measures undertaken by Fabian Thommen in 2018. He migrated the work to a current web server with an old PHP version 5.5.9 and made the following changes:

- He replaced the Google library from 2004 with a Google Custom Search API from 2018.
- The database commands in PHP had to be replaced in order to be compatible with newer PHP versions.
- Configurations like the database connection and the Google API keys were moved to a configuration file in order to reduce maintenance and increase security.
- Security was enhanced in parts. One big security risk is the passing of variables such as user data from the client to the server. There are different methods, how a browser client can send information to the web server. In *TraceNoizer* the user variables were passed as register_globals to the PHP script. This method is insecure, as the input can be easily manipulated. Thus, from PHP 5.4.0. on only the GET and POST methods are possible. Fabian Thommen adapted the scripts accordingly.
- *TraceNoizer* used the rainbow library in order to analyse the text of the websites generated by the user's search. The latest rainbow version dates from 2002 and its binary was compiled for a 32bit operating system. To run the 32-bit binary on a 64 bit operating system, the library lib32z1 had to be installed.

The *significant properties* of the work were partly preserved: With this migrated version, clones can be generated, and the user can login and delete his/her clone. The scripts and programming languages are only so much changed that they function. The look and feel of the website stay the same, even if the website is viewed on a current browser. This will not be the case in the future, as frames are not supported in HTML5. However, this can be solved with a browser emulation. Almost all the significant properties are respected with one exception: the clones are faulty. They are so faulty, that they do not fulfil the purpose of pretending to be somebody's homepage (text with visible html tags, missing images). Most of them do not contain images and the sub-pages made of keywords are also missing. This can be caused by malfunctioning of the rainbow library, or the fact that the structure of web pages has changed so drastically since 2004, that the extraction of sentences and images

does not work properly. It is also possible that the clone engine has never functioned as intended and has always produced a certain number of faulty clones.

Regarding *sustainability*, the migration strategy does not yield the best results. It is able to deal with *software obsolescence and changed network protocols* such as HTML or the Google API, but at the expense of changes in the code. In addition, these changes have to be repeated every few years to keep up with changes in web technology. Hence migration does not stabilize software complexity but rather enhances it. *Maintenance* will be high in order to alleviate security risks but also to clean up the database periodically. The *scalability of the preservation strategy* is relevant for the House of Electronic Arts, as they host other internet-based artworks. However, as each works needs an individual web server software environment, the migration strategy for this work does not scale.

It can be summarized that the migration strategy met the significant properties of the *TraceNoizer* website, but not the ones of the clones. Its biggest shortcoming from the perspective of long-term preservation is the fact that it needs to be repeated regularly. Serious Internet security concerns and an expected high amount of maintenance add to the disadvantages of this strategy.

5.7 Preservation Version “Emulation”

As an alternative, another preservation strategy was tried by emulating the Conservix CD of *TraceNoizer* using the University of Freiburg’s Emulation-as-a-Service (EaaS) framework (Rechert and et al. 2019). EaaS provides users with convenient access to emulators via their web browsers. A curator can ingest a digital object, configure the right emulator and its settings, and allow any user to start the configured emulation environment. By default, each user is presented with a fresh emulation environment as configured by the curator and several users can use different sessions of the same emulation environments at the same time. Alternatively, however, emulation environments can also be "snapshotted" by users or curators, conserving their current state including any manipulations by the user.

Previously, EaaS concentrated on emulating single (unconnected) environments, e.g., a single preserved work of digital art as an archived CD-ROM image, which needs to be run on a Windows-95 environment with an installed Macromedia Flash player. As is outlined in this work, preserving single works as stand-alone entities is not sufficient for many works as their significant properties are realized through and depend on the

combination of several systems. Thus, it is necessary to regard the whole ecosystem as one (connected) preservation environment.

To facilitate the emulation of a connected environment, a recent addition to EaaS allows to create a virtual network, which operates on the Ethernet layer (Davoli 2005, 213). The virtual network is represented by a URL for identification and for access control. Via this URL, the virtual network can be reached from the Internet using the WebSocket protocol over HTTPS. HTTPS/TLS encrypts the traffic and, thus, shields it from malicious access from the Internet, while the usage of the WebSocket protocol (as opposed to direct TCP/IP) shields the Internet from the emulated environments (which malicious users of the EaaS system might otherwise abuse to perform, e.g., DDoS[43] or spam attacks on the public Internet). At the same time, it introduces a layer of emulated Ethernet traffic on top of the EaaS web API and, thus, shields the EaaS host system from Ethernet traffic between the emulated environments. Inside the WebSocket connection, Ethernet frames from the connected emulation environments are prefixed with a simple two-octet big-endian length header (the same format as used by the VDE 2 library [VDE-community n.d.]). Firstly, the described concept allows to connect multiple emulation environments. The emulation environments can either be specific to the preserved digital object (e.g., in a multi-machine system consisting of an application server and a database server) or generic emulation environments can be combined ad-hoc. This approach allows to easily reuse emulation environments by a curator or a user without any necessary special knowledge.

In the case of *TraceNoizer*, its Conservix-CD version was emulated by the EaaS framework using the QEMU emulator (s. Figure 29). In a second connected emulation environment, a web browser that was current at the time of the creation of the artwork (2004) was started. As a further step, other emulated environments containing web browsers could be built to allow users to examine *TraceNoizer* (and any other digital objects). As *TraceNoizer* was originally built at a time in which optimizing websites for specific web browsers (and build upon their non-standardized features) was prevalent, this could be essential to fully reproduce the original performance of the artwork as perceived by different users at that time.

43 DDoS: In a Denial of Service attack the perpetrator tries to flood a web service with superfluous requests. In a Distributed Denial of Service attack the perpetrator uses different computers to perform these attacks. See (Khosrow-Pour 2013, 170).

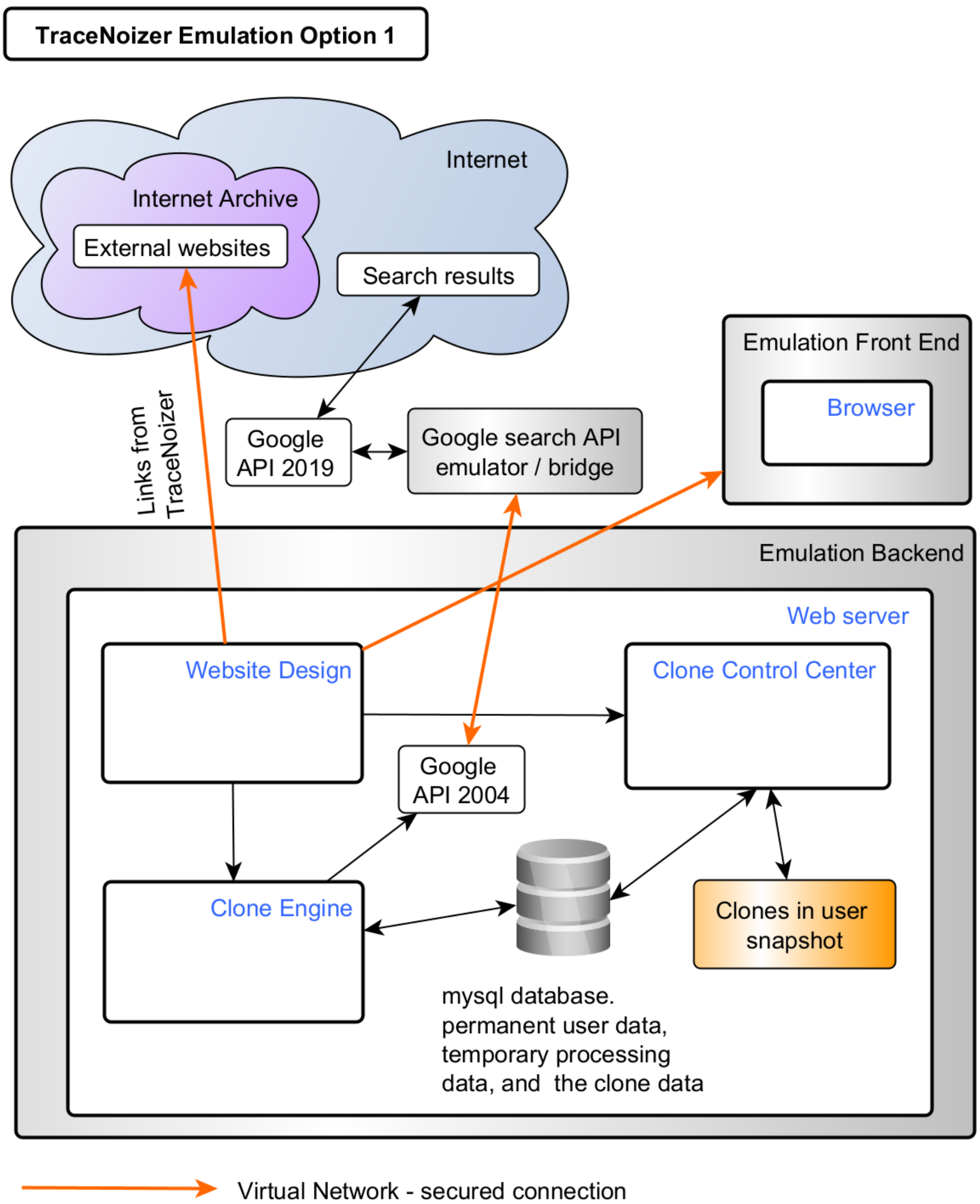


Figure 29: Emulation option1 for *TraceNoizer*. The emulated *TraceNoizer* backend interacts with the current live Web. Generated clones are not exposed to the live Internet.

Secondly, the approach of virtual networks allows to offer additional services in the network. E.g., it is currently already possible in EaaS to connect the virtual network to the current live Internet, and thus, allow a connected user to access current websites. To fully recreate the original *TraceNoizer* performance, instead of allowing access to the live Web, an archived version as, e.g., provided by the Internet Archive[44] could be used (s. Figure 30). This would effectively operate as a transparent proxy[45], operating at either the DNS and/or

[44] The Internet Archive is an American institution harvesting the internet and making the harvested websites publicly available. Website of the Internet Archive see (Internet Archive n.d.).

[45] Also known as interception proxy, see RFC 3040, section 2.5 (Cooper, Melve, and Tomlinson 2001).

HTTP layer. The proxy could either be preset by a curator or configured by the user to serve the Web as archived on a specific date. This would allow, e.g., to retroactively analyse the behaviour of *TraceNoizer* at different points in time.

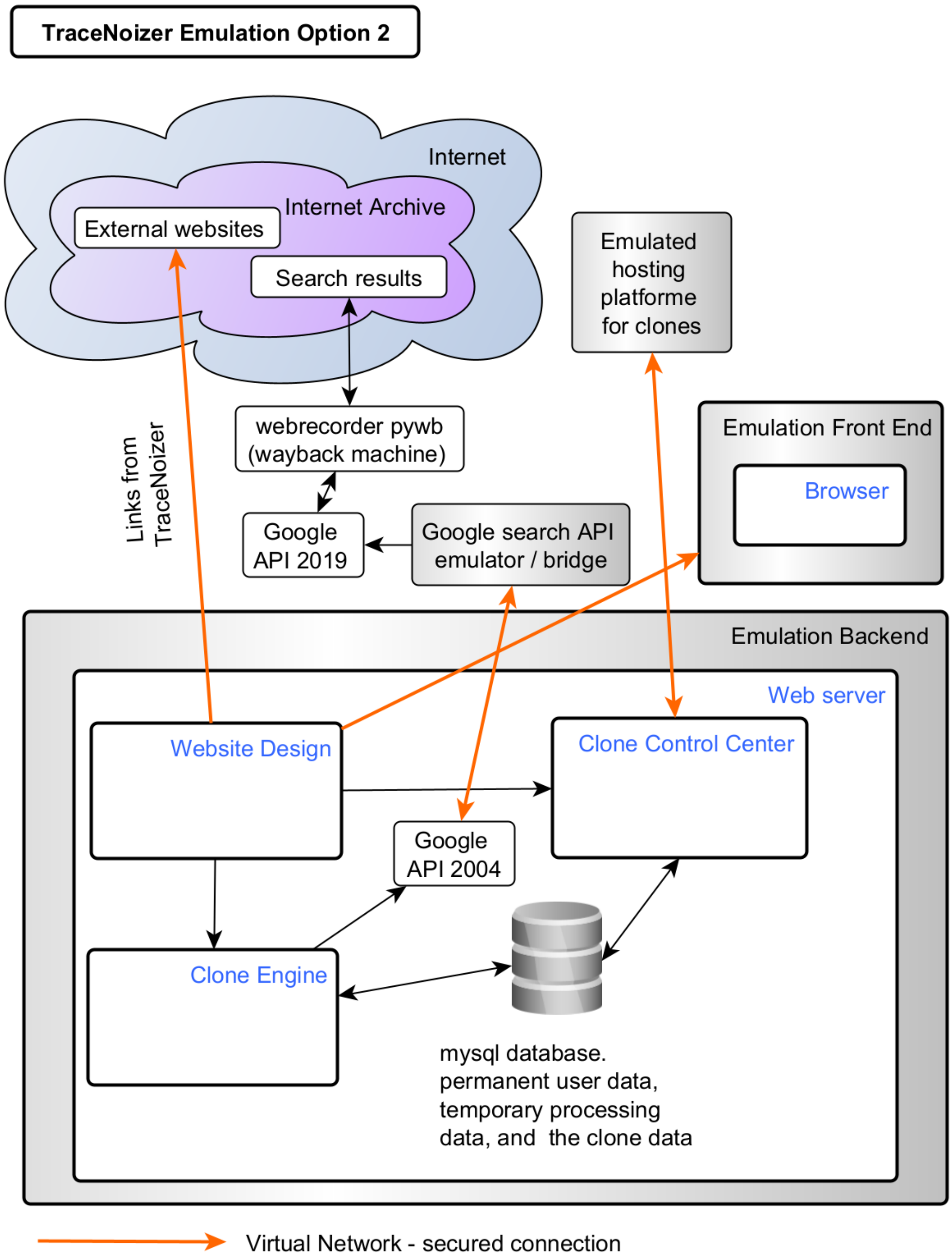


Figure 30: Emulation option2 for *TraceNoizer*. The Internet Archive's Wayback Machine[46] is used to let the emulated *TraceNoizer* backend interact with an archived version of the Wolrd Wide Web. Generated clones are exposed to the live Internet.

[46] The python wayback machine (pywb) is based on the Internet Archive's wayback machine. Source code see (Webrecorder community n.d.).

The virtual network cannot only connect emulated environments but also allows (via eaas-proxy) applications from outside to access the services provided in the virtual network. It, therefore, allows to map and forward an external TCP port to an internal IP address and TCP port. The termination of this connection can either occur in the public Internet (at an individually assigned network port for each user session), at the user's computer on localhost (by downloading and running a eaas-proxy binary), or, in the future, directly in the user's browser (using ServiceWorkers, see (Archibald and Kruisselbrink 2022)). It allows the user to view *TraceNoizer* in their current web browser, "breaking out" of the emulated environment and makes operation like interacting with the website content, copying contents, or even deep linking to contents much easier. The user is, though, still protected from any malicious emulation environments by their web browser, and the operator's system is protected by the virtual network, which is directly connected to the emulated environment but separated from the actual host system.

A further problem, which can be solved by the presented approach, is the usage of ancient Google Search web APIs (utilizing SOAP) for the *TraceNoizer* system. Google has stopped supporting this API, but it can be emulated in a virtual network environment and, consequently, allows the *TraceNoizer* environment to remain unchanged. The same approach is applicable for *TraceNoizer's* storing clones on web hosts which have long ceased operation.

5.8 Discussion of Preservation Strategies

For internet-based artworks, that originate in the fast-changing technical and social environment of the internet, it is necessary to abstract from the concrete technical setup and formulate more high-level principles. The following three principles can be used as guidelines for maintaining the functionality of internet-based art- works in a new networked environment:

a) Reconstruction of the artwork environment: Part of the old environment is recreated in order to adapt to the old artwork and allow better interaction without having to change the original artwork too much.

b) Bridging: A bridge can be built between the old artwork and the new environment. This can be achieved by encapsulating the artwork and then providing an interface to translate the input/output between the old and the new environment. It could also be

achieved by adapting the code of the artwork directly to communicate with the new environment, which would correspond to a migration.

c) Reinterpretation of the artwork: The artwork, or parts of it, could be recreated by different means, such as new platforms or new technology, in order to adapt to the new environment. For instance, *TraceNoizer* could be recreated on the Facebook platform, by cloning Facebook accounts instead of websites.

Web archiving of *TraceNoizer*'s external links is an example for the reconstruction of the artwork environment. Broken external links of *TraceNoizer* could be downloaded from the Internet Archive and saved in a protected environment. This protected environment is stable and in control of the conservator.

The emulation of the Google Search web API as described in the previous chapter is an example of the bridging strategy, as it allows to use the Google Search Engine. The exchange of the Google Search API with the current API as done in the migration of *TraceNoizer* is also seen as bridging from the "old" work to the new environment.

The above-mentioned principles are carried out by applying a combination of reprogramming, migration, emulation / virtualisation, and web archiving. As the examples showed, these principles can be applied to different elements of the website such as web browser, web server, or to its external dependencies such as parts of the World Wide Web environment and external web services.

External web services are used more and more in internet-based artworks. External web services can be preserved or handled as described by Dragan Espenschied and Klaus Rechert in *Fencing Apparently Infinite Objects* (Espenschied and Rechert 2018, 2). They are suggesting the mirroring of the web service, a stub interface that has a reduced functionality instead of the web service, and the recording of network traffic of a web service for a limited number of queries. All three proposals correspond to the principle of reconstruction of the artwork environment mentioned above. If reconstruction is used as a strategy to preserve web services instead of bridging with an API, it is likely that a compromise with the functionality or authenticity of the work has to be accepted.

In order to link different emulated elements of the internet-based artwork, the University of Freiburg enhanced EaaS by enabling to connect these emulated environments within a virtual network. This allows a great flexibility in finding tailor-made preservation solutions. In addition, the user can login in this network and save his/her data by taking emulator snapshots. This has the advantage that the web server including database can be reset for maintenance purposes without losing user data.

The networked environment does not exist for contained software-based artworks. For contained software-based art, hardware can play a much more important role than for internet-based art. Hardware can be compared to an external dependency, as specific peripherals cannot be easily rebuilt, and their production is dependent on the production company. Hence, for contained software-based art, the bridging principle is important when replacing old equipment with new equipment. Emulation of input/output devices, as for example bridging the tracking sensor of *Horizons* (2008), is an example for this (see section 4.6). Running a new piece of equipment within a case of an old device is comparable to the reconstruction of the environment of an internet-based artwork. Using the old case simulates the old device. Hence, it can be concluded that contained software-based art relies on the same preservation strategies as internet-based art with the exception of web-archiving.

The following section will investigate *TraceNoizer*'s display history and whether its short-term curation had an effect on its long-term preservation and vice versa.

5.9 Presentation Forms and Presentation History of *TraceNoizer*

In addition to the preservation strategies discussed above, the different forms that *TraceNoizer* has taken when presented are also important when considering the relationship between its short-term curation and long-term preservation. As such, this final section focuses on *TraceNoizer*'s display history.

During the years of its creation, 2001–2004, *TraceNoizer* was presented at festivals. 2001 saw the artists present the work at the Viper 01 Art Festival in Basel, Switzerland, and the READ_ME Festival 1.2 in Moscow, Russia; in 2002, *TraceNoizer* went to the Hartware Medienkunstverein (HKV) in Dortmund, the *Transmediale.02* festival in Berlin, the *Digital-is-Not-Analog* festival in Campobasso, Italy, and the *Open Source Art Hack* festival at the New Museum in New York, USA; in 2003, the work was shown at the *ArtBit Collection* in Tokyo, Japan[47]. At the *Transmediale.02*, the artists presented *TraceNoizer* on a stage, standing in front of a projection (see Figure 31).

[47] Source for the list of these exhibitions: (Lee n.d.b).

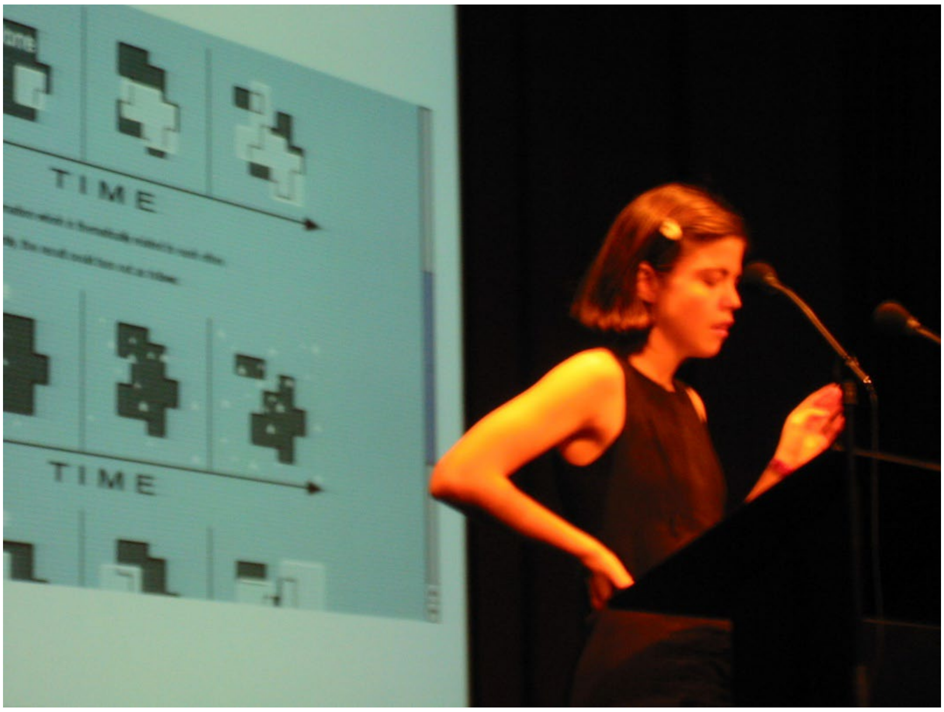


Figure 31: *Transmediale* 2002 in Berlin. *TraceNoizer* presented by a member of LAN, Annina Rüst. Source: project archive received by the artists, HEK collection.

At the New Museum, New York, *TraceNoizer* was shown on computer terminals in the festival's Media Lounge (see Figure 32).

Figure 32: Zenith Media Lounge, New Museum, New York, USA in 2002. Source: (Lee n.d.b)

Elsewhere, LAN created a kind of information desk where people could create clones and receive printouts of their clone profile in the form of business cards (see Lee n.d.a] and Figure 33).

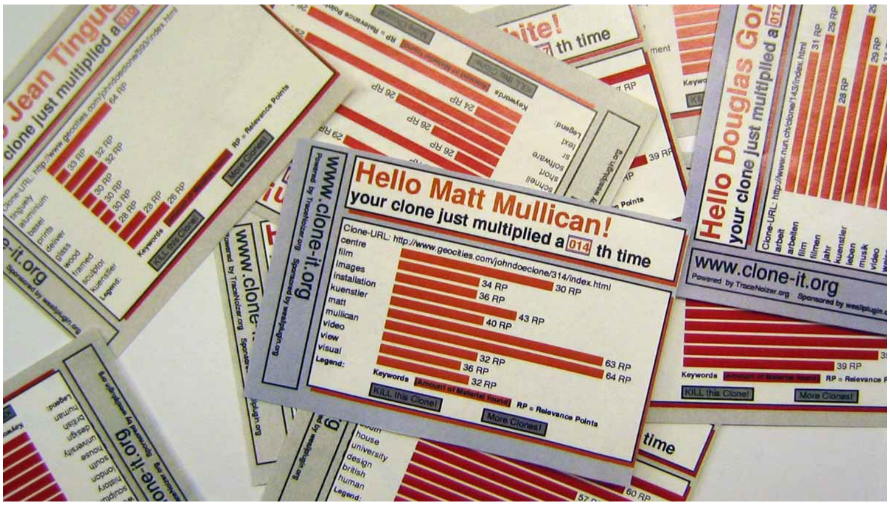


Figure 33: “Business cards” for users of the clone engine. *Clone-it* project at LISTE art fair, Basel, 2001. Source: (Lee n.d.a).

LAN called this project *Clone-it*, and used the same clone engine as it had for *TraceNoizer*. In 2001, *TraceNoizer* was presented at the *READ_ME Festival 1.2* (READ_ME Festival 2002) which in addition to being an art festival in the traditional sense, was also an early online platform for software-based artworks.

Only much later did *TraceNoizer* again become part of online exhibitions. For the 2018 exhibition *Right to RE-* (Jennings and Boas 2018), *TraceNoizer* was shown alongside ten other digital artworks that addressed the issue of user autonomy on the internet. The URL given for *TraceNoizer* did not, in fact, lead to the work itself, but to documentation of it on Marc Lee’s website (Lee n.d.b), as *TraceNoizer* was no longer online then. In the *We-Link2.0: Sideways* online exhibition of 2020 (Zhang 2020/2021), the browser window took on the appearance of a computer desktop from the 1990s and provided historical context for the work (see Figure 34). Each artwork was located in a folder on the computer desktop. Upon opening a folder, a text window appeared, containing a curatorial text about the artwork and a URL. The artwork then opened in its own browser window.

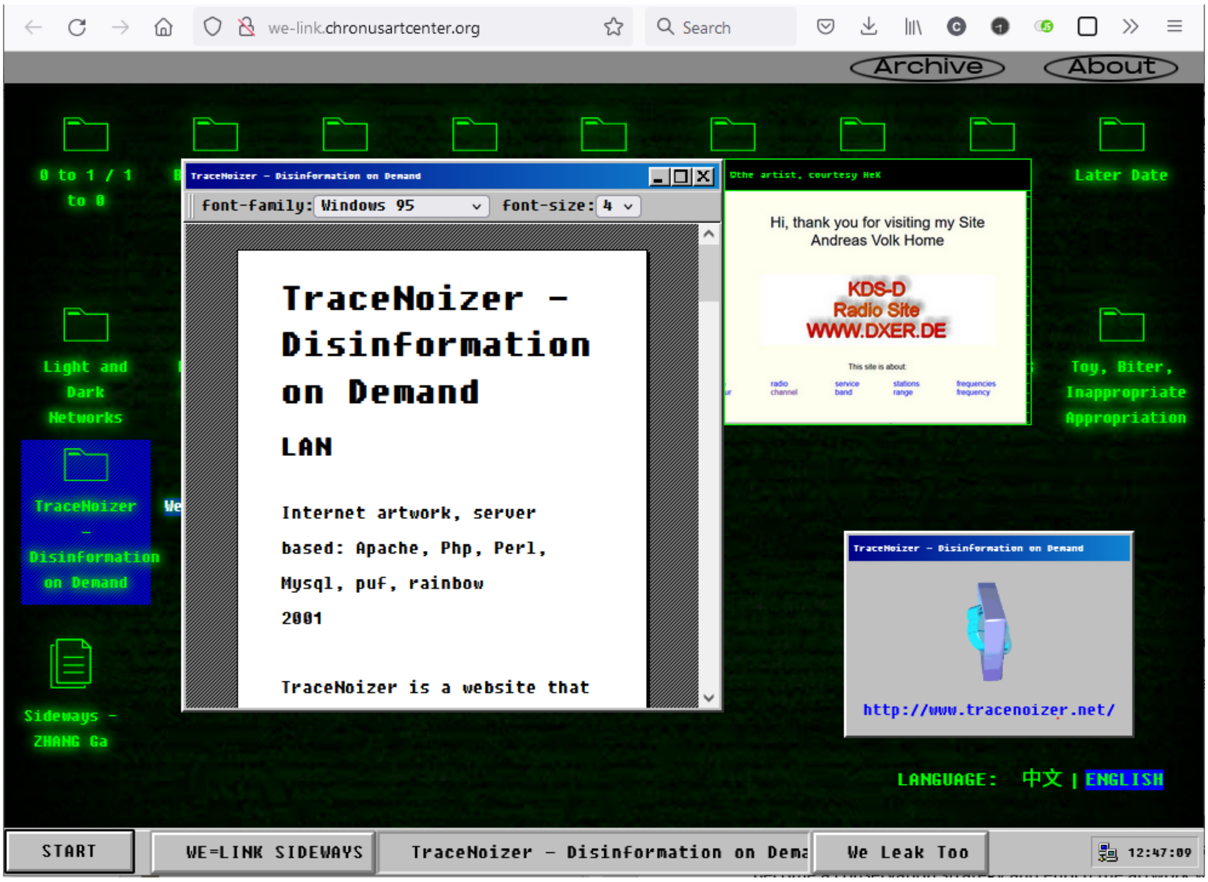


Figure 34: *We-Link2.0: Sideways* exhibition in 2020. The link in the window on the bottom right led to a new browser window containing *TraceNoizer*.

In 2022, the House of Electronic Arts created an online exhibition to present its collection. Each artwork, including *TraceNoizer*, was shown within an iframe[48] with a fixed size on the online exhibition platform (see Figure 35).

[48] "An HTML iframe is used to display a web page within a web page." See: (W3 Schools n.d.)

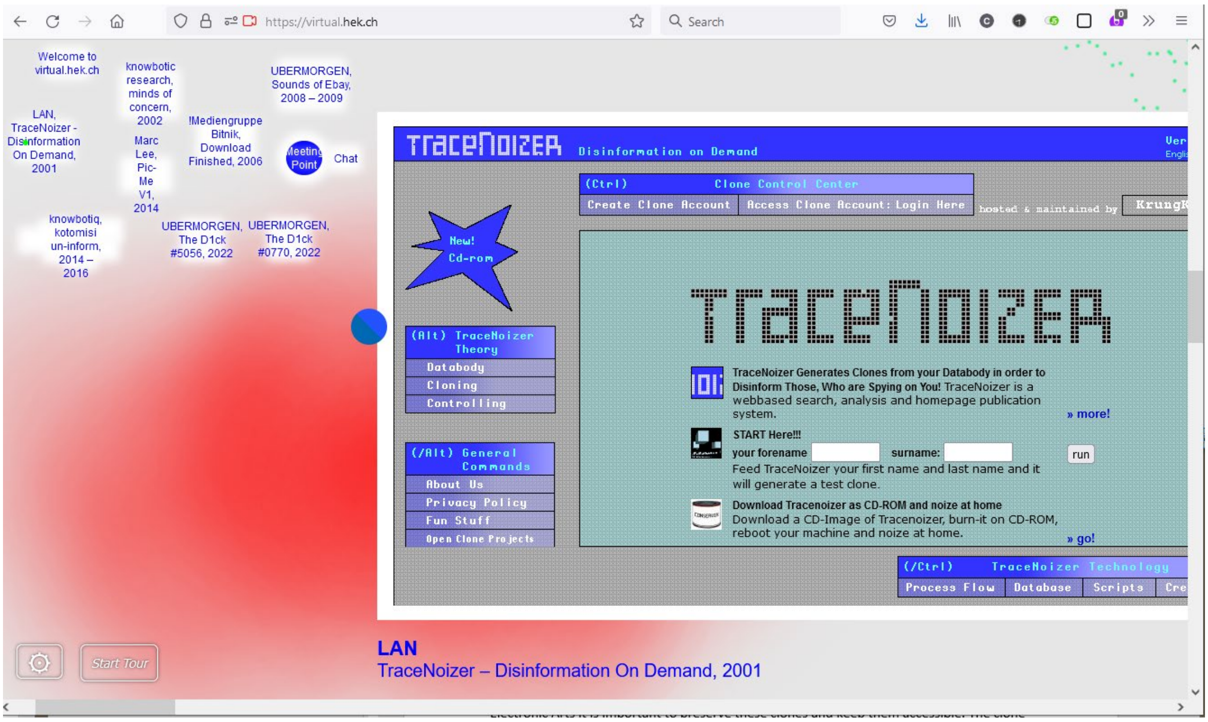


Figure 35: Online exhibition *From the Early Web to Web 3.0* in 2022 (Himmelsbach 2022). *TraceNoizer* is presented within an iframe.

Whilst these three online exhibitions and *TraceNoizer*'s presentation at festivals yielded very different performances of the work, the *Tracenoizer* website itself was not adapted for exhibition purposes. The only exception here was *Right to RE-* exhibition, which presented documentation of *TraceNoizer* as it was no longer online at that time. The various exhibition designs and topics, as well as the use of specific hardware, wall prints, and other paraphernalia all contributed to a variety of presentations of *TraceNoizer*, without having to change its software or adapt preservation strategies. It is worth noting, too, that the preservation strategies examined in the previous section did nothing to limit the presentation options described here.

5.10 Conclusions for Sustainable Preservation of Internet-Based Artworks

This chapter has explored the preservation strategies and presentation options for the internet-based artwork *TraceNoizer* (2001-2004), and has examined their impacts on the work's long-term preservation. Following on from the discussion of preservation strategies in section 5.8, we may conclude that the more external dependencies can be eliminated, the more sustainable a preservation strategy is. As such, a reconstruction of *TraceNoizer*'s web environment through a combination of web archiving and the creation of a virtual network is

the best strategy, as it replaces the evolving web environment with a stabilised archive. In addition to the reduction of dependencies, the reduction of maintenance and security aspects are dominant considerations when assessing the sustainability of preservation strategies for internet-based artworks. As was the case with *TraceNoizer*, a migrated website is unsafe and prone to web attacks. Therefore, it can prove more economical to restart an emulation regularly than to maintain the server, which can take an hour or more weekly. The isolation of several internet-based artworks from each other – all with different needs and software environments – is another potential argument for emulation and/or virtualisation. In the future, a fair compromise between the reduction of maintenance, security considerations, and functionality will be crucial for the preservation of internet-based artworks.

TraceNoizer's presentation on a variety of digital platforms and its incorporation as part of a physical installation did not impede its long-term preservation. On the contrary, the range of presentations shows just how many presentation possibilities exist without having to compromise on long-term preservation. Moreover, if – hypothetically – the preservation strategies explored in this chapter had been applied to *TraceNoizer* when it was exhibited, they would not have constrained presentations of the work as delineated in section 5.9.

6 Analysis and Discussion[49]

Over the course of my research, I conducted two case studies to get a deeper insight into the conservation of software-based art. Each represents a typical example of such art: *Horizons* by Geert Mul is an interactive, generative, and graphically intensive installation, whereas *TraceNoizer* by LAN is an internet-based artwork. Both are subject to some of the most common challenges faced when conserving software-based art: for *Horizons*, the obsolescence of equipment, and for *TraceNoizer*, the changing internet environment from Web 1.0 to Web 2.0. In both case studies, I applied two preservation strategies and compared them for their long-term sustainability. This chapter will discuss the findings of my case studies in light of the theoretical framework developed in Chapter 3, to evaluate and explain their results.

Section 6.1 reformulates the core problems faced by conservators of software-based art based on my experiences of the two case studies. As such, it extrapolates certain general conservation challenges for software-based art from the specifics of the cases. In section 6.2, I apply the theoretical framework laid out in Chapter 3 to both case studies, with the purpose of assessing the long-term sustainability of the various preservation measures. Section 6.3 adapts the existing definitions of conservation strategies to extend their applicability beyond the domain of fine arts conservation to the broader sphere of digital preservation, as practised in libraries and archives. To adapt the artwork to an exhibition context, in section 6.4 I provide options for long-term conservation to accommodate short-term curation. Section 6.5 proposes guidelines for a software-based artwork acquisition strategy that would allow as many conservation options as possible to be kept open. This would improve sustainability in the long term. In section 6.6, I discuss the institutional requirements necessary for the sustainable conservation of software-based art.

6.1 Core Set of Problems when Conserving Software-Based Art

Software-based art's cycles of conservation are short, and range from a few months to a few years between interventions. This has to do with the ever-changing socio-technical

---

[49] This chapter is based on
Roeck, Claudia. 2023. "Languages of Conservation: A Comparison Between Internet-Based Art and Built Heritage." In Conservation of Contemporary Art: Bridging the Gap Between Theory and Practice, edited by Renée Van de Vall and Vivian van Saaze. Studies in Art, Heritage; Law and the Market 9: Springer. https://link.springer.com/book/10.1007/978-3-031-42357-4

environment of software-based artworks. Internet-based artworks are particularly illustrative of this genre's dependency on the socio-technical environment, as they often depend on other websites, web services, or platforms. As the internet is very dynamic, these websites and web services change regularly (in a process known as content drift), change location, or even disappear altogether (in a process known as link rot)[50]. Moreover, internet technology and culture change, too. The ongoing transition from Web 1.0 – the early World Wide Web, made up of primarily personal, static websites – to Web 2.0 – where platforms and web services dominate – is one example. Furthermore, web browsers are subject to frequent changes such as security updates, and adaptations to the development of browser languages such as Javascript and HTML, a cause for websites functioning incorrectly.

TraceNoizer, the internet-based artwork discussed in Chapter 5, went offline in 2012. As such, all that remained was a bundle of source code, torn out of its environment. The only way to see it was through a Linux Live CD, which reconstructed the work's hardware and software environment by emulating a web server and web browser. The internet environment, necessary for generating clones, is accessible only if the Linux Live CD is run on a computer with internet access. However, creating clones is now impossible with the Linux Live CD, as it uses a search API that is incompatible with today's search engines. In addition, the free webpage hosting services that hosted the clones generated by *Tracenoizer* no longer exist. Website culture itself has changed too: nowadays, users express themselves through social media profiles rather than personal websites. Also worth mentioning is the huge shifts that website aesthetics have undergone, such that the clones generated by *TraceNoizer* do not resemble current web pages and could hardly be mistaken for them.

Computer-based installations have different dependencies to internet-based artworks. Whilst they do not rely on internet protocols and linked websites, they do rely on hardware and software environments. The hardware environment, consisting of computers and input and output peripherals, is not as stable as it may appear at first. CPUs and GPUs produce heat and can fail, especially in exhibition settings where computers are switched on for weeks or even months and are sometimes placed inside small boxes with limited ventilation. In the long term, failing hardware must be replaced with newer models that have different

---

50 "Two characteristics of the dynamic nature of the Web are link rot and content drift. Link rot occurs when links break over time – a detriment we have all encountered numerous times, and that has been studied and quantified abundantly. Content drift happens when content at a given web location changes over time" (Sompel, Klein, and S. Jones 2021).

software requirements to the older models which they replace. In other cases, hardware is specific and cannot be replaced. Compatibility between hardware drivers, the operating system, and the application software is crucial for the hardware to function properly. This is also true for media carriers, hence why disk images must be read via an emulator for the right computer architecture and disk drive.

Horizons, discussed in Chapter 4, is an installation that comprises a computer-generated video. Other dependencies are the piece's tracking sensor and its video projector. As described in section 4.3, the sensor solution was not stable when *Horizons* was first exhibited in 2008, which pushed the artist to find a more stable sensor that he then used for *Shan Shui* in his 2016 retrospective exhibition in Schiedam. In 2021, the sensor needed to be replaced for *Shan Shui*'s permanent installation in Dortmunder U (see section 4.9). Unfortunately, an identical model was no longer available, and a new sensor library had to be used. As such, changes in the artwork's socio-technical environment – such as the manufacturer no longer producing the old tracking sensor – forced the artist to use updated equipment and find a way to make this compatible with the software artefact.

The changing environments of both *Horizons* and *TraceNoizer* have either caused the artworks to break or forced the artist/s to replace equipment with different elements to those originally used, leading to adaptations in the software. Whilst updated equipment was initially welcome in both cases as it improved the artworks, these adaptations later became a burdensome necessity. It is necessary, therefore, for long-term preservation to find a way to deal with an artwork's changing environment.

Another core problem that affects the conservation of software-based art is the disappearance of knowledge about interacting with the artwork's interface. Due to the rapid development cycles of hardware and software, software user interfaces and hardware input devices, such as gaming joysticks, change quickly. For instance, the design of *TraceNoizer*'s user interface does not meet today's requirements. In 2001, website elements that automatically adapted to mobile phone screens, as in the case of responsive web design, had not been developed. Users of *TraceNoizer*, therefore, need to know that the work is better viewed on a computer screen than on a mobile device.

Furthermore, it might be difficult to understand an artwork's content after a period of time in which the historical context has changed. *TraceNoizer* is an example of just such a work. When they were developing the piece, LAN's concern was that users did not have total governance over their own internet presence. For example, opinions expressed in an online forum could not be removed or changed years after they had been posted. Nor could

texts and images published by others about oneself be deleted. *TraceNoizer* was a response to this situation, enabling the user to blur the traces of their online existence by adding and mixing new web pages that looked similar to the actual ones but contained nonsense.[51] Although this problem of persistent unwanted content on platforms still exists in 2023, social media users have become accustomed to it and adapted their behaviour accordingly. Instead, the critical focus has shifted to questioning the power and surveillance of commercial social media platforms.

This section has illustrated how the technological environment, user knowledge, and current context of software-based artworks shift quickly, posing a challenge to their legibility over time. The key question, therefore, is how conservation can handle this rapidly developing socio-technical environment sustainably. The following section explores how my theoretical framework may contribute to achieving this goal.

6.2 Contribution of the Theoretical Framework to Sustainable Preservation Solutions

The theoretical framework developed in Chapter 3 proposes an expansion of conservation theory to encompass artworks that are based on software as a medium. Its goal is to support the sustainable preservation of software-based art, beginning by interpreting the art object as an artefact embedded in an environment that is constantly changing, as opposed to considering it to be an isolated object. The framework sees the artwork and its environment as a technosystem, a term coined by Feenberg, to describe a system that includes "disciplines and operations associated with markets, administrations, and technologies" (Feenberg 2017, x). Indeed, the conservation of artworks is influenced by markets, administrations (understood to mean institutions in this dissertation), and technology. Feenberg takes a constructivist approach to technology, considering it to shape knowledge and culture, which then in turn shape technology. My contribution combines the technosystem theory with Dourish's approach to the materiality of the digital, which highlights those properties of digital artefacts that shape and limit user experience. Reflecting on the materiality of the digital is a "way of assessing the relationship between the social and the technical" (Dourish 2017, 38/190). The third pillar of my theoretical framework consists of sustainable preservation criteria, applied to assess and select

[51] The artists called these newly generated web pages clones.

conservation strategies for software-based artworks. Until now, conservation theory prioritised criteria such as reversibility or minimal intervention. However, criteria that support long-term preservation were missing. This section of my analysis therefore reflects on how the theoretical framework might contribute to the sustainable conservation of software-based art as it faces the challenges described in the previous section. For each of the three dimensions of my framework laid out above, I begin by describing its relevance to software-based art conservation. Second, I indicate how this is evidenced in my two case studies. Third, I show how this dimension may be meaningfully integrated into conservation practice. Finally, I close each subsection with a short conclusion on the salience of that component for the long-term sustainability of software-based art conservation.

Digital Materiality According to Dourish

An early step in any conservation process is an analysis of the artwork and its context. An assessment of the meanings of the material provides important information from which preservation goals can be determined, and about how much a material might be changed without negatively impacting these meanings. Although the software is often not visible, it determines the behaviour of an artwork. The materiality of software can be understood as the limitations it may have, and how it does the things that it wants to achieve (Dourish 2017, 12/190). Secondly, the materiality of the digital can be interpreted as "the gap between what is denoted [code] and what is expressed" in combination with hardware (Dourish 2017, 24/190). Thirdly, an assessment of the socio-cultural background of software enables the conservator to classify the use made of it by the artist, as well as its rarity. This socio-cultural assessment includes information about how widespread a piece of software was, what it was commonly used for, and which user groups mainly used it. As software-based art preservation is a relatively new field in conservation, an assessment of the historical and cultural meaning of the software material is not a common routine for conservators. Such knowledge is, rather, currently to be found in the academic domain of media studies.

The programming languages used for Geert Mul's *Horizons* are Ruby and C. Mul asked Carlo Prelz, a programmer working within the Dutch art and open-source community,[52] to write the code in close collaboration with him. The C program sick.c is

[52] V2, Rotterdam, is a space where the use of open-source software in arts is encouraged. An example of this culture was the workshop "Creative Kernels," held in 2000 (V2 Rotterdam 2000), in which Geert Mul and Carlo Prelz participated..

required to interpret the tracking sensor data, and the Ruby programs are responsible for generating the video and audio based on the tracking sensor data. Ruby and C are both still active programming languages, and there has been no reason to replace them with newer ones. In general, C is often used to program device drivers, operating systems, or other programming languages, whilst Ruby is more abstracted (higher-level) than C and typically used to program applications. Neither the historical[53] nor age[54] value of the software artefact is high yet, as the artwork was last updated in 2016. As to whether the code should be migrated/updated, or whether it should run in a virtualisation, this question stems more from a sustainability point of view than from a materiality point of view. It could be argued, though, that because the programs are custom-made for *Horizons*, and given that the way in which Ruby and C are used bears witness to a specific time and programming culture, the programs should not be updated if possible (in line with the conservation principle of minimal intervention).

For *TraceNoizer*, the artist group LAN wrote the scripts themselves. *TraceNoizer* is programmed in PERL, PHP, and HTML based on a variation of LAMP[55], an open-source web server programming stack typically used in the 2000s. The programmers used PERL to parse the search results and then reassemble the text for the clones, as PERL is a language that is well suited to parsing texts. PERL is still actively used, but it is less popular than in 2002. To build up the dynamic website, LAN used framesets, an HTML tag which is no longer supported in HTML5. *TraceNoizer* is an example of 2000s web technology, and therefore has historical value. As a result, replacing the original programming languages with more current ones would remove the artists' imprint, or "handwriting". The conservation of the work aimed to preserve the website's dynamic nature and its generation of clones. This led LAN to do a minor update of the PHP version, so as to decrease the risk of hacking attacks, and to update the Google Search Engine API to make this possible. From a conservation perspective, this was considered a minimal intervention. Updating to the newest PHP and Linux version would have posed a considerable risk to the website's functionality, and would have required Fabian Thommen to debug it, which was not a feasible option.

To evaluate the significance of an artwork's digital materiality in conservation practice, I suggest using a matrix developed by the built heritage specialists Kuipers and de

53 See (Riegl 1982, 1) and section 2.5.
54 See (Riegl 1982, 4) and section 2.5.
55 Software stack for web servers. Linux, Apache, MySQL, PHP/Perl or Python. See (Udoh 2009, 1110).

Jonge (Kuipers and Jonge 2017, 87). They use a matrix to evaluate every layer of a building, thereby creating a basis for making decisions about conservation measures. Each building's layer has its own materiality. The rows of the matrix that I am suggesting consist of the components and functional layers of a software-based artwork broken down to its digital material. The columns of this matrix consist of Riegl and Appelbaum's values: aesthetic value/behaviour, use value/functionality, age value/newness value/rarity, historical value, and artistic value, plus a socio-cultural value which I had to add. An example of such a table, filled in for the *Horizons* and *TraceNoizer* case studies, is to be found in Appendix A.1. This method also helps the user to identify the artwork's different layers of meaning and thus contributes to a better quality of conservation.

By researching and documenting the meaning and socio-cultural value of the software used in an artwork, a conservator may become more aware of the software as material, and more attached to it. This then leads to conservation strategies that stay closer to the material, such as emulation or migration, based on the principle of minimal intervention. Valuing software materiality in this way could be criticised for limiting conservation options. However, in many cases it is not the artists who choose to adapt their artworks; it is a necessity borne of the rapid obsolescence of technologies which can compromise the artist's intended meaning. Furthermore, software-based artworks are rare examples of objects based on old software that are still in use. The fact that there are not many art, design, video game, science, or industry museums that preserve and research software artefacts is itself a good argument to try and preserve original code where possible.

Artwork Analysis According to Feenberg's Technosystem

The second dimension of my theoretical framework derives from Feenberg's technosystem, which sees the design of artefacts to bring together layers of function and meaning (Feenberg 2017, 32). As described in section 3.7, all software-based artworks consist of functional layers and building blocks. Functional layers are hierarchical. They typically occur in networks or in a computer as a software and hardware stack. A building block is a part of an artwork that can be treated with its own conservation strategy independently from the rest of the work. It is informed by dependencies and can be layered itself (see Figure 14).

In *Horizons*, identifying layers such as SDL[56] and OpenGL[57] led to an understanding that the artwork could be encapsulated using paravirtualisation. These layers abstract the access to the video card and thus enable paravirtualisation to enhance the rendering performance of the virtualisation. Paravirtualisation worked very well for the video output, but the sound output was not satisfying. The conservation strategy must therefore be tested and the result compared with the artwork's significant properties, even if the strategy is expected to work based on an analysis of the work's layers and dependencies. In this sense, important details such as interfacing with devices, the emulation of input and output devices, and computer performance, can make all the difference for a conservation strategy.

For *TraceNoizer*, it was more important to identify building blocks than layers, as the rendering performance was less of an issue. In this case, the building blocks "web server", "web browser", "search API", and "generated clones" were identifiable (see Figure 29 and Figure 30). For each of these blocks, several preservation strategies were conceivable. As described in Chapter 5, the web server could either be migrated or emulated, the search API could be migrated to a current search API or emulated/bridged, the web browser could be realised either as a current web browser or a so-called browser emulation, and the clones could either be stored on the web server or on an emulated external hosting platform. Identifying building blocks such as these can support a conservator to find individual solutions for them, and provide a certain flexibility by combining them.

In addition to an artwork's functional layers and building blocks, its environment is also conceptually important for its conservation. This insight leads to the recommendation that an artwork's environment must be considered in its preservation.

Horizons is not networked and has no wider digital environment to be preserved. However, it relies on video projectors with a specific aspect ratio and a tracking sensor, all of which are dependent on their production environment. This environment is outside of the conservator's sphere of influence[58] and cannot usually be stabilised or adapted, although an emulator is available that could be seen as a form of stabilisation for the computer hardware.

---

56 SDL (Simple DirectMedia Library): low-level access to audio, keyboard, mouse, and graphics hardware. See section 4.6.

57 OpenGL (Open Graphics Library): interface between video card and graphics software. See section 4.6.

58 There is at least one exceptional case where the production of obsolete equipment was resumed due to the demand for such equipment: the startup Colorvac in Germany resumed the production and repair of CRT monitors for the first time since about 2010. Some video-based artworks made since the 1960s require a CRT monitor for presentation, as either the CRT image quality, the CRT technology, or the sculptural

TraceNoizer's closer environment is made up of links to external websites. Its clone engine accesses the Web, which could be considered *TraceNoizer*'s wider environment. Either the live World Wide Web's state of continuous change is accepted as a fact, and *TraceNoizer* continues accessing it as it did when it was online, or one attempts to stabilise the environment. As described in section 5.7, it would be possible to have *TraceNoizer* access a specific year of the Internet Archive, instead of the live World Wide Web, to create the clones and access the external links. As the Internet Archive captures live websites at specific dates and freezes them, the historical context that it could provide would support a reading of the artwork in a specific way.

From my experiences with *TraceNoizer* and *Horizons*, it can be concluded that identifying an artwork's functional layers, building blocks, and environment all support the process of deciding on appropriate conservation strategies. The following section discusses the sustainability criteria that must be at the heart of any selection process for these possible strategies.

Sustainable Conservation Strategies

The third dimension of my theoretical framework is the sustainability criteria for conservation strategies. In contemporary fine arts conservation, reversibility – or retreatability – and minimal intervention are important principles. However, as my research has shown, the rapid conservation cycles of software-based art, the fragility of its digital materiality, and its ties to its socio-technical environment all make additional sustainability criteria necessary. In this section, I will examine how the sustainability criteria formulated in section 3.1 apply to my case studies, and whether they can be applied to conserve software-based art more generally.

In section 3.1, I argued that conservation strategies are sustainable if they ensure access to and the readability of the artwork in question; if they minimise the effort needed for later conservation cycles; and if they are scalable; if the network of care stabilises the definition of the artwork's significant properties; and if this network transparently documents the measures and decisions taken. I will now go through and discuss these sustainability criteria one by one. From my original list as detailed in section 3.1, sustainability criterion d) – that conservation strategies must allow future generations to

---

quality of CRT cubes is significant to them. The socio-technical environment therefore adapted to the demand of collectors and institutions.

preserve the artwork in different ways – will be discussed separately in section 6.5, as this has to do primarily with the question of acquisition. Sustainability criterion g) – the role of the institution in ensuring the sustainability of conservation strategies – is discussed in section 6.6, as it is not a property of conservation strategies but rather a prerequisite for them.

According to criterion a) as defined in section 3.1, one of the primary goals of conservation strategies is to ensure that artworks are accessible and readable. Simply storing the software in a digital repository and the physical components in an appropriate storage facility, without the possibility of bringing the artwork out for exhibition, does not meet these criteria. To make sure that an artwork such as *Horizons* remains accessible requires the active management of preservation cycles, including monitoring and maintaining its equipment. For the internet-based artwork *TraceNoizer*, accessibility means that the user can access it online, which implies the regular maintenance of the web server and proper management of its preservation cycles. Readability has to do with the work-defining properties of an artwork. The readability of *Horizons* is ensured as long as its significant properties as described above are guaranteed. For *TraceNoizer*, readability must be improved if the clone engine ceases functioning as it should. Due to its self-documenting nature (several pages on the *TraceNoizer* site describe the idea behind the artwork), *TraceNoizer* is indeed readable with there being little need for additional explanation either on an institution's website or in exhibition documents.

Criterion b) states that preservation strategies should lower, or at least avoid increasing, the effort required in later preservation cycles. This is crucial, as there have been some cases of internet-based artworks, such as Marc Lee's *TV Bot*, that have undergone regular migrations without there being any reduction in the work required for the next preservation cycle. Most institutions cannot afford to hire a programmer to migrate artworks every few years. Stabilising source code complexity is, therefore, one means of stabilising, or even decreasing, future conservation efforts, and can be achieved through abstraction and generalisation, principles in computer science that are embodied in emulation (see section 4.4). In contrast to emulation, the strategy of migration increases source code complexity (Belady and M.M. Lehman 1985, 253). Another notable way of reducing future preservation efforts is to stabilise the artwork's environment by capturing and reconstructing it. This reduces the artwork's structural complexity, and thus the work needed to maintain it and its external dependencies. The more external dependencies can be internalised in this way, the lower the future preservation efforts.

For my case study of *Horizons*, an emulation was first envisioned, as this would have made the artwork completely independent from its hardware. In the end, however, emulation was not feasible for performance reasons. As a compromise between performance and independence from hardware, therefore, I tested paravirtualisation. However, paravirtualisation was unable to output a sufficiently good sound quality for *Horizons*, making migration the only option. As a result, the artwork's significant properties could only be met by updating the software to an up-to-date operating system.

For *TraceNoizer*, I identified four building blocks and an environment in the previous section, and tested individual conservation strategies for each of them. I will now apply the second criterion of sustainability to Google Search API as the building block, and the World Wide Web as the environment.

For the obsolete, dysfunctional Google Search API, three solutions present themselves: reconstruction, migration, or emulation of the search engine. Reconstructing a search engine is very demanding technically and financially, and is therefore unfeasible for the House of Electronic Arts. Whilst it would make *TraceNoizer* independent from an external search engine and give the House of Electronic Arts full control of its search API, it would also bring its own challenges, such as maintenance needs. Migration (updating the search API command in *TraceNoizer*'s source code) is not demanding technically, but has to be repeated regularly and constitutes a change of the source code. Bridging the obsolete search API would not change the source code, but would have to be adapted regularly to the new API (see Figure 29 and Figure 30 in section 5.7). Both migrating and bridging the search API commands have the disadvantage of it being impossible to know in advance when the search API requires an update, hence why the functioning of the website would need to be monitored.

TraceNoizer's environment, the World Wide Web, is significant for both its external links, and the clone engine that searches the Web and uses these search results to create new clones. If *TraceNoizer*'s clone engine runs on archived pages from the Internet Archive, its environment is stabilised, and therefore no new web technology will pose problems for its functioning. This is more sustainable in the long term than accessing the live Web, which evolves continuously.

Sustainability criterion c) requires conservation strategies to be scalable. If the Museum Boijmans Van Beuningen had other software-based artworks in need of virtualisation based on the same virtualisation platform, then the virtualisation applied to *Horizons* would be scalable. However, the second strategy that I applied to *Horizons* –

migration – is not scalable, and must be applied and repeated each time anew. For *TraceNoizer*, scalability is a given. The House of Electronic Arts runs *TraceNoizer* in a container,[59] which means that the work shares infrastructure with and is maintained alongside other internet-based artworks. As such, scalability, though relatively easy to implement for certain types of artworks such as those that are web-based, is often not possible for others with specific hardware requirements. Moreover, scalability constitutes a strong case for storing software-based artworks together: if storage infrastructure can be shared by multiple artworks, it can be managed and maintained more sustainably.

Criterion e) requires the definition of an artwork's significant properties to be stabilised. Whilst the definition of work-defining properties may fluctuate at the beginning of an artwork's life cycle, it should be stabilised after a few iterations, resulting in a consensus around the work's significant properties (those to be preserved). This means that once the significant properties converge, they should be changed as little as possible over the course of preservation cycles. This reduces both the need to create new versions and the effort required for conservation and documentation.

Horizons has changed little since its creation in 2008. As described in depth in section 4.3, I regard the artwork's source code as a significant property, as it is the most precise description of the work's video and sound output (although, according to Mul, *Horizons* could theoretically be coded differently and still achieve the same effect). The sound is something of a pink noise, and its quality is determined by the quality of the work's thirteen WAV samples. For the video, the colour space (RGB), resolution, and aspect ratio of the video frame (3072 by 768 pixels) are considered significant. On this topic, the artist has expressed some flexibility in light of new video projector technology, accepting that future projectors with higher resolutions will be able to interpolate missing pixels.[60] In hindsight, we know that Mul accepted the creation of a video with a higher resolution and a different aspect ratio for *Shan Shui*'s permanent installation in Dortmund in 2021 than the piece had had for its original 2013 installation (see section 4.9). However, the Dortmunder U commissioned the artwork for a specific space, which is why he was able to modify it in this way. I also considered the behaviour of the video pattern to be significant, and therefore defined a speed range for the horizontal movement of the image columns based on video

[59] A container is an operating system virtualisation, see glossary.

[60] Interview with Geert Mul about the conservation of *Shan Shui* and *Horizons*. Interviewer: Claudia Roeck (Mul 2017b).

captures from *Horizons* running on a slower computer in 2008 and on captures from 2016. Without such measuring, the speed difference would not have been noticeable.

In 2008, at the exhibition in the Museum Boijmans Van Beuningen, the artist used a thermal camera as a tracking sensor for *Horizons*. As it did not work reliably and several problems occurred during the exhibition, Mul sought a more stable solution. For *Shan Shui*, in 2013, Mul stabilised the sensor input by using a very sturdy laser sensor that he was satisfied with and that we also used to test *Horizons* (see Figure 17 and Figure 23). I cannot say whether the artwork's significant properties changed through the stabilisation of the tracking sensor, as I never saw the version with the thermal camera. Considering the sensor history, however, I would define its reliable functioning as significant, and not the brand or model of the sensor.

Generally, knowing how narrowly to define an artwork's significant properties, so that any changes to it can be properly identified and when to set or reset these properties, is a challenge. The rapid development of peripherals complicates the stabilisation of significant properties. The network of care must try to imagine the variability of the artwork in question, even though they may never have seen an actual instantiation of it. As neither the artist nor the conservators are able to predict all of the situations that an artwork might encounter, it can take several instantiations of the work before all parties can agree on what constitutes an acceptable variability of its properties.

The sustainability criterion f), the transparent documentation of conservation measures, is a matter of course in the field of conservation. On one hand, such documentation enables later conservators to identify previous conservation measures, which is particularly important in the digital realm, as changes of code cannot be recognised if they are not recorded as such. On the other, it also allows conservators to understand the reasoning behind certain measures. Both of my case studies were documented with the help of a version control system that allows comments to be left for every change of source code.

To conclude this section, my primarily qualitative sustainability criteria invite conservators to make decisions about conservation strategies with a long-term perspective in mind, which is an important achievement. However, reducing structural complexity and stabilising source code complexity are not easy to achieve in practice. Compromises often have to be accepted, as non-functional requirements such as performance and security factors, or an artwork's significant properties, need to be considered. Therefore, the conservator should be aware of the different preservation strategies' nuances, as they offer different degrees of sustainability and options for retaining the work-defining properties of

the artwork in question. Stabilising these significant properties is a challenge due in part to the fast-changing hardware and software environment, as well as the fact that software-based artworks are often experimental in nature.

Limitations of the Framework

Whilst the theoretical framework helps potential conservators to analyse an artwork, including its environment, and identify the most suitable conservation strategies for long-term preservation, it cannot predict technological or social developments in the future. Instead, it tries to make the artwork independent from such developments. Applying the theoretical framework can reduce the number of necessary conservation interventions, or their complexity, or both. If possible, these interventions are moved to the interface between artwork and environment, instead of targeting the artefact alone.

The theoretical framework is suitable for the conservation of artworks in a narrow sense. For strategies such as reinterpretation and the continuation of processual artworks, different sustainability criteria apply. The strategy of reinterpretation does not need to reduce the complexity of existing artworks, and is not bound to their materiality. Instead, it must apply the principles of software sustainability that are relevant to the creation of new software, such as those suggested by the Software Sustainability Institute (Software Sustainability Institute n.d.).

The theoretical framework addresses conservation strategies that can be carried out at the level of software. My research does not look at extending the life expectancy of hardware, although this can be important for certain software-based artworks. Such conservation strategies would be situated between the realms of preventive conservation for physical objects and the maintenance of electronics.

Although the theoretical framework does not directly address such principles of conservation ethics as minimal intervention, respect for the material used, and retreatability or reversibility, these nevertheless apply. Therefore, the ethical dilemma of either preserving an artwork's integrity, particularly if it is heavily damaged, or facilitating readability by restoring it (Janis 2005, 138) is not resolved here.

Section 6.2 discussed both the contributions and limitations of my theoretical framework. The following section proposes certain adaptations to the definitions of conservation strategies. The proposals come as a consequence of my having applied the theoretical framework, and are intended to harmonise conservation strategies between digital preservation and the contemporary conservation of non-digital art.

6.3 Conservation Strategies: Adapted Definitions and Introduction of “New” Strategies

The previous section explored the ways in which the theoretical framework can contribute to the analysis of a software-based artwork in the context of its conservation. It also verified the applicability of the sustainability criteria for conservation. Systematic applications of the framework would require consistent descriptions of the relevant preservation strategies. This would remove any contradictions between the definitions as they exist in the digital preservation field and in the conservation of contemporary art, and thereby facilitate the proper application of the strategies. As mentioned in section 2.4, conservation strategies as defined for contemporary art by the Variable Media Network are not wholly compatible with those as defined in the field of digital preservation. As software-based art exists at the intersection between contemporary art and digital preservation, it is highly important to articulate definitions that are suitable for both fields. Therefore, to merge these two approaches, I suggest the following adapted definitions, which I will discuss in the following sections:

- The strategy of emulation in digital preservation will be expanded to include other types of abstractions. They will be referred to as encapsulation strategies, as this term is also significant in contemporary (non-digital) art conservation.
- The strategy of reconstruction will be introduced as a conservation strategy. The strategy of emulation as defined by the Variable Media Network corresponds to this reconstruction strategy.
- The strategy of documentation will be introduced as a conservation strategy.

My definitions of the storage strategy and the migration strategy do not differ significantly from those used in digital preservation and by the Variable Media Network. The new set of conservation strategies, therefore, consists of storage as a basic strategy alongside migration, encapsulation, reconstruction, and documentation[61].

Encapsulation (Emulation in Digital Preservation)

I use encapsulation as an overarching term to refer to the emulation strategies employed in digital preservation, as it can be used unambiguously in both digital

[61] The strategies of reinterpretation and continuation of an artwork complete the set of conservation strategies. However, they require different sustainability criteria and are therefore not researched here.

preservation and contemporary art conservation. Encapsulation protects a software artefact and its software environment from change by encapsulating them within an additional layer.

In the digital realm, different types of abstraction layers abstract the computer architecture, input and output peripherals, and network protocols. In an emulation or virtualisation, a layer abstracts the computer architecture, encapsulates the software environment of a digital artefact, and makes it possible to run the software artefact unchanged on a new computer.

Instead of encapsulating an artwork's entire software environment, it might be sufficient to have a local encapsulation such as a bridge[62] (or adapter, or interface) to just one peripheral. Such a bridge between the drivers of the old and new peripherals has a similar function to encapsulation: it isolates the software artefact from the new peripheral through abstraction. Instead of changing the artefact, the bridge is adapted to the new peripheral.

Another form of abstraction that isolates the artwork from changes are abstractions of network protocols. They enable the creation of a virtual network by integrating virtual machines/emulations in a network independently from an underlying physical network such as the internet. The virtual network can be set up so that external IP addresses for the internet are mapped to internal ones for the virtual network. This allows software (for instance, software installed on an emulated web server) or users to keep accessing nodes in the virtual network with the original Uniform Resource Identifier (URI). The idea of a virtual network is to internalise any external dependencies in combination with other conservation strategies. The network traffic of such a virtual network is encapsulated, hence why the overarching term encapsulation is suitable for this kind of abstraction, too.

The advantage of encapsulation is that it removes the artwork's software from needing to be touched, because the additional layer represents an abstraction of the original computer hardware or network protocol. Encapsulations not only comply with the minimal intervention principle, therefore, but are especially useful if the software is complex and custom-made. Furthermore, we must assume that collaboration with the artist as described by Hannah Hölling (H. B. Hölling 2013, 44) will phase out in the long term. It can be difficult to find a programmer who has the patience and dedication to deal with code written by someone else. It is much easier, however, to find a programmer who can set up and maintain an up-to-date emulator. Unlike migration, encapsulation does not need to be

62 A bridge in this dissertation is different from a network bridge as defined in computer science.

repeated, only continued by maintaining the emulator. It neither reduces nor increases source code complexity, though it does add the complexity of an additional layer. Naturally, this strategy is only feasible if an emulator for a specific piece of hardware or protocol exists. Moreover, introducing an abstraction layer can cause performance problems. Quality problems can also arise, when for example hardware is not emulated well enough or abstracted too much (this may affect the video or sound card, for instance). There is also the risk that an emulator will not be supported in the future.

For *Horizons*, the emulation strategy needed to be adjusted to tackle performance issues. With paravirtualisation (video card virtualisation) and virtualisation (CPU access), performance could be improved. For the tracking sensor, I suggested building a bridge between the old tracking sensor and the new one. This is necessary if they do not use the same data protocol. By building a bridge, the software that includes the driver for the old tracking sensor would not have to be changed (see Figure 21 in section 4.6).

TraceNoizer is installed in a Linux container, which is a process virtual machine (Smith and Nair 2005, 13). Process virtual machines are based on the abstraction of the Application Binary Interface (ABI), and remain dependent on the computer hardware as well as the Linux kernel. However, as containers are much lighter in terms of storage space and processing power than emulations, the House of Electronic Arts decided to use a container system for the art websites that they host. In the case of a major Linux kernel update, the hardware would have to be emulated and the container would have to run within the emulator (Rechert et al. 2016, 228–29).

In general, it can be said that the more a virtualisation has access to hardware (such as the CPU or a video card), the better its performance will be, and the more it will depend on this hardware. Thus, emulators, classic virtual machines, and containers all have different levels of hardware dependencies. It is also worth noting that I do not consider application virtual machines that are installed on top of operating systems such as the Java Runtime Environment sustainable conservation strategies, for they constitute an additional dependency. A knowledge of these different types of virtual machine is useful when looking for compromises between an independence from hardware and other non-functional requirements such as performance.

The *TraceNoizer* case study also provides an example of a virtual network: as described in section 5.7, and illustrated in Figure 30, the web server emulation, the web browser emulation, the Internet Archive, and the bridged Google search API can all be connected to form a virtual network. A fake DNS server carries out the mapping of URLs

for the internal IP addresses of the network nodes. This virtual network insulates *TraceNoizer* from changes in its World Wide Web environment (as described in section 6.1). By creating a more stable environment, it extends the duration of *TraceNoizer*'s conservation cycles, and reduces the constant effort needed to adapt the work's software to its changing network environment.

In this section, I proposed the term encapsulation as a broader and more unequivocal alternative to emulation outside of digital preservation. Furthermore, I expanded the encapsulation strategy to include not only virtualisations of computer architecture, but also other abstractions such as bridges for peripherals and virtual networks. This encourages conservators to differentiate between different types of virtualisations as a means of adapting the encapsulation strategy to non-functional requirements.

Documentation

The Variable Media Network, though it is a major reference for contemporary conservation in terms of strategies for time-based media art, does not include documentation in its list of conservation strategies (see section 2.4). However, documentation is an important strategy for software-based artworks. It conserves an artwork's historicity, and for artworks based on obsolete technology it is sometimes the only strategy available besides reinterpretation (Espenschied quoted in (Dekker and Giannachi 2022, 13)]). I therefore propose documentation as a fully-fledged conservation strategy. Documentation as a strategy means that a documented version of an artwork is presented in an exhibition, instead of the fully functioning artwork. The strategy consists of two sub-strategies: curatorial capture, and the presentation of archival and documentary material. The difference between these two sub-strategies is that curatorial capture directly intercepts the artwork's data streams, whereas archival and documentary material is created with external devices such as cameras or sound recorders, or through people describing their experiences of the artwork. I will begin by discussing curatorial capture, then briefly touch upon the presentation of archival and documentary material.

Curatorial capture means either capturing a software-based artwork's inputs and then running it from those captured inputs, or capturing its outputs and presenting these instead of those that would be generated in real time. This capture can occur at different levels, such as the tracking sensor data of *Horizons* before or after it has been transformed via a device driver. If the data is captured at a higher (more abstracted) level, it will be less hardware-dependent.

Whilst captures have the same materiality as the channel that they were recorded from, they do not usually include the logic of the software. Hence, through capturing, the artwork loses parts of its performativity and interactivity. Although this may compromise the artwork's defining properties, replacing dynamic content with captured content is often more sustainable. The sustainability of documentation as a strategy depends on the technology used to present the captured content. There are three possible ways of integrating captures with the artwork.

Firstly, the captured output replaces the artwork's dynamic software components. This can be achieved by replacing the dynamic output of a generative software-based artwork such as *Horizons* or *Shan Shui* with a static screencast (video). In terms of sustainability, this is a very good solution. In terms of work-defining properties, however, this would not be acceptable in the case of either *Shan Shui* or *Horizons*, as the visitors would no longer be able to interact with the video by moving around. For documentation to be a sustainable strategy, the captures must be presented with sustainable (that is, simple and ubiquitous) technology such as web browsers or video players[63] that are hardware-independent. As such, captures used to create a virtual reality documentation of an artwork would not be sustainable, as virtual reality headsets are hardware-dependent, and the maintenance of VR engines is complex.

Secondly, captures are made available (based on simple and sustainable presentation technology) in addition to the software-based artwork. For instance, in *TraceNoizer*, the artists made the clones that users created in 2001 available on the *TraceNoizer* website. The clones that were outputs of the clone engine of *TraceNoizer*, are made of HTML, and need a simple web browser to be interpreted. Therefore, although *TraceNoizer*'s clone engine does not currently function, the user can browse the archive from 2001 and see what the clones looked like.

Thirdly, the captured input is fed into the artwork's software, instead of the user input. An example of a captured input can be found in the *TraceNoizer* case study (Figure 30, section 5.7). It was suggested to run the clone engine on the Internet Archive and not the live World Wide Web. Captured websites from 2001-2004 would be used as the input for *TraceNoizer* in the hope that the clone engine would work better on old website captures (from Web 1.0, before social media) than on today's Web 2.0. Linking to a web archive

[63] 360-degree video formats and players are becoming more widely used, hence the probability that they become more sustainable.

reconnects the internet artwork with its former environment whilst introducing a new, but relatively stable, dependency on a web archive.

Although this capturing strategy does compromise an artwork's work-defining properties, it may be the only feasible and most sustainable preservation strategy, as the captured artwork may cease to function due to technological obsolescence, and/or because repeated migrations risk changing it too much. Most importantly, the captured, static content is (usually) easier to preserve in the long term than dynamic content, as the artwork is stabilised by the reduction of dependencies and complexity. For instance, if the captures replace external dependencies such as content supplied by web services, this reduces the risk of further functionality losses in the future.

Snapshots of classic virtual machines are another form of capturing. They capture the disk and memory of a virtual machine executing the processes of a software-based artwork. However, I do not consider these virtual machine snapshots a sustainable conservation strategy: their execution depends on the computer hardware that the snapshot is made on. Furthermore, they only work with the virtual machine that they were made from. Hence, whilst they might be useful for maintenance, they cannot be considered a sustainable documentation strategy.

I will now address the second type of documentation strategy, which involves archival and documentary material to represent the artwork. Such material describes the artwork more broadly than captures of its input and output channels. It is usually made with devices that are not part of the artwork, such as external video or photography cameras, and often includes documentation of the artwork's environment, such as its hardware and the space around it. Text documents and physical components help to contextualise the artwork. Furthermore, archival material may have been recorded with a purpose other than conservation. Similar to curatorial captures, there are multiple presentation options here: either simply replacing the artwork with the archival material, or presenting the archival material in addition to the artwork. Presenting the archival material is only sustainable if sustainable presentation technologies are used. The use of such material as a conservation strategy is based on Hanna Hölling's concept of the artwork as an archive (H. B. Hölling 2015, 87).

If the artwork evolves, new captures or documentary material may become necessary, though when and how often the artwork is captured or documented is a curatorial decision. With captures and documentation, the conservator and curator create a narrative for the artwork, and attempt to circle in on its identity. Each set of captures tells a story that

is subjective to a certain extent; in this, they can be compared to memories that contribute to the identity of a person or a group. As Dusan Barok et al. explain with reference to SFMOMA's use of Wiki software for the documentation of contemporary art:

> [Documenting and editing artworks on a Wiki] also sheds a new light on the often-repeated statement that art installations only exist when they are installed. Compiling and editing relevant documentation not only prepares us better for their live episodes in galleries, but helps us treat them as being present and alive in institutional memory. (Barok, Noordegraaf, and de Vries 2019, 15)

This memory does not have to remain within institutional boundaries, however, and can also serve the memory of the public. Thus, curatorial capture and the presentation of archival material can be conservation strategies, and enrich the artwork with context as a replacement for lost functionality or interactivity.

TraceNoizer's clone engine does not function properly, and the user can be prepared for this incomplete experience by being informed that the artwork's age is the reason for certain dysfunctionalities. This information was added to the description of *TraceNoizer* on the HEK collection website (House of Electronic Arts Basel 2019) and to the *TraceNoizer* website itself. Therefore, if documentation is adopted as a conservation strategy, visitors or users need to be aware of this so that they can form their expectations accordingly, and understand any dysfunctionalities of the artwork. Although software-based artworks can be seen as "self-documenting devices" (H. B. Hölling 2013, 37), the user does not often have access to the source code, and will need additional information to understand potential gaps in artwork function.

Thus, I consider the documentation of software-based art to be a fully-fledged preservation strategy, that can either be combined with the existing software artefact or replace it entirely. Due to the reduction of external dependencies, this strategy is highly appropriate in view of the artwork's long-term preservation. Whilst it often restricts the artwork's interactivity, it preserves its historicity and context well. Documentation of the documentation would then be required if the artwork's functionality becomes compromised.

Reconstruction

The Variable Media Network uses the term emulation in two ways: sometimes to refer to encapsulation as described in the previous section, and sometimes to refer to reconstruction (see section 2.4). To clarify this double meaning, I propose reconstruction as

a separate conservation strategy. The aim of reconstruction is to achieve the same look or function of an art object, without necessarily using the same building material (software) or structure (inner logic). Reprogramming a software functionality with a different programming language or software constitutes a reconstruction. This strategy was already mentioned in the DOCAM project (see section 2.4). Reconstructions may be necessary when the artwork, or part of it, is lost (see, for instance, Wijers 2011, 83]), or to replace external dependencies.

As already described in section 6.2, the example of *TraceNoizer* offers use cases for reconstruction. Theoretically, the search engine used in *TraceNoizer* could be reconstructed to avoid dependency on the Google search API. However, such a reconstructed search engine would be limited to the functionalities needed for *TraceNoizer* and would probably provide inferior search results than a commercial search engine. Miksa et al. (Miksa, Mayer, and Rauber 2015, 78) describe how this kind of mocked-up web service could work. As the institution would be in charge of it, there would be no unexpected changes to the API, which would make for heightened sustainability. However, the institution would then have to guarantee the maintenance of the reconstructed search engine. Another use case for a reconstruction would be to create a virtual network based on harvested (captured) websites used as inputs for the clone engine instead of the live World Wide Web (see (Roeck et al. 2019, 189). Such a case would combine the strategies of encapsulation and documentation (curatorial capture) to result in a reconstruction.

Reconstructions should be as independent from external technical developments as possible, and should aim to stabilise the artwork by reducing external dependencies. This therefore reduces maintenance in the future. The choice of preservation strategies applied to the reconstructed piece in a second stage, such as migration or encapsulation, also has an impact on long-term preservation. Reconstruction brings with it the risk that certain less obvious features (for instance so-called Easter eggs[64] on websites or in games) are overlooked.

To summarise, I propose reconstruction to differentiate it from the strategy of encapsulation. For reconstruction to be sustainable as a strategy, it must internalise external dependencies, which can be achieved by combining it with the strategy of documentation.

[64] Easter eggs are hidden features in a video game or on a website (Butterfield, Kerr, and Ngondi 2016, 178).

Combination of Strategies

Conservation strategies can be combined in different ways. They can be nested as in the case of *Shan Shui* and *Horizons*, where the paravirtualisation of hardware was combined with a migration of the operating system (see Figure 36).

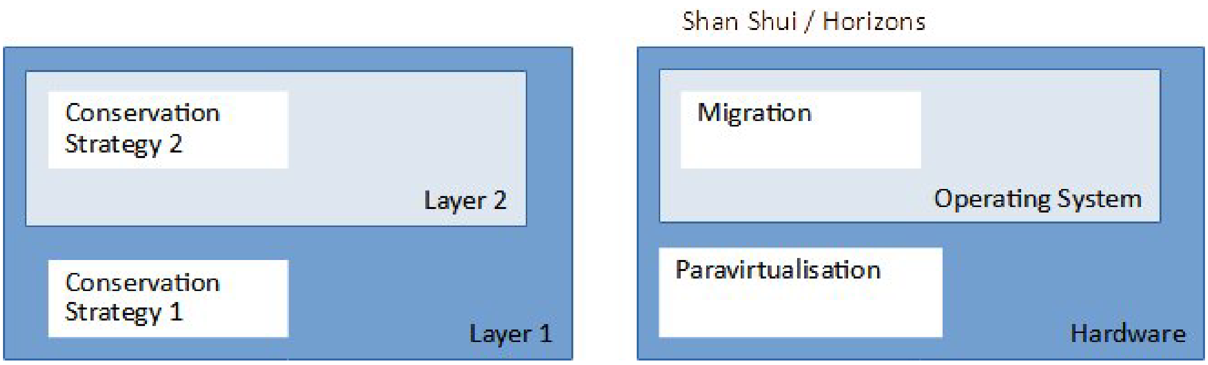


Figure 36: Nesting of conservation strategies for *Shan Shui* and *Horizons*.

Another example of nesting conservation strategies can be seen in the case of *TraceNoizer*. A virtual network (following the encapsulation strategy) was combined with an emulation of the back end, a browser emulation, a bridge to Google Search, and the Internet Archive (see Figure 37).

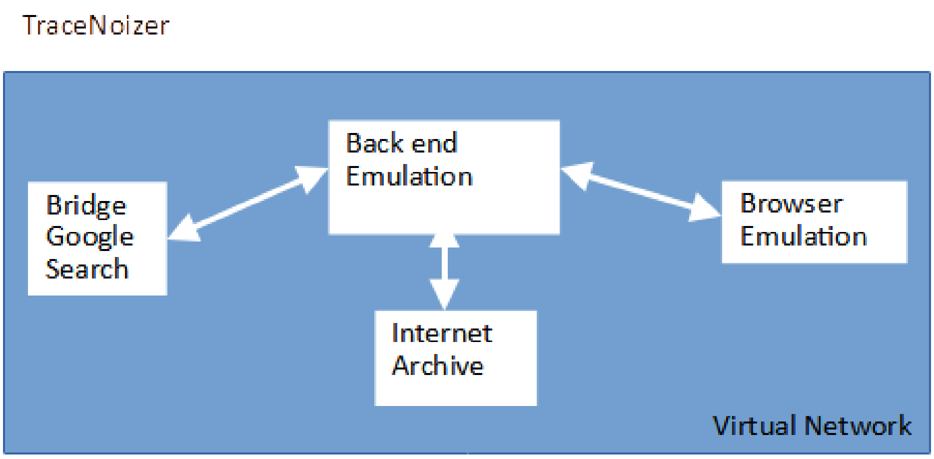


Figure 37: Nesting of conservation strategies for *TraceNoizer*, “Preservation Version Emulation” (see section 5.7).

Another way to combine conservation strategies is by applying a separate strategy for each building block in parallel. For web-based artworks such as *TraceNoizer*, the backend can be migrated, and a modern web browser chosen for the front end (see Figure 38).

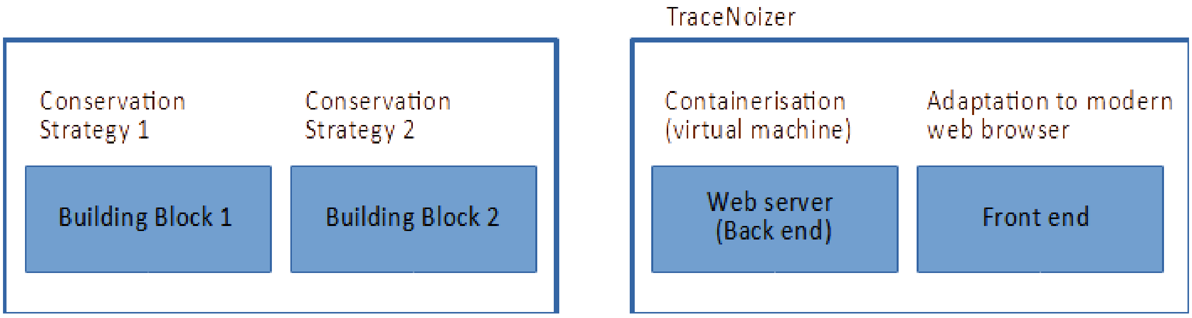


Figure 38: A separate conservation strategy for each of *TraceNoizer*'s building blocks. "Preservation Version Migration" (see section 5.6)

As a strategy, storage is an exception to the two types of combination described above. It is a basic strategy upon which all other strategies build, and is carried out as part of the acquisition process as well as after each conservation cycle. In contrast to all other strategies, storage keeps the object in a frozen, deactivated state. The artwork's components are not connected to each other physically or digitally, and might be stored in different locations. Unlike physical objects, additional conservation strategies are – where possible – carried out on duplicates of the stored digital objects, creating new versions.

Strategies must be handled consistently when combined. This is particularly true where previous conservation measures have been undertaken, such as local updating of code or links, as applying conservation strategies on top of these may prove challenging. An example of this is *Minds of Concern* (2002) by Knowbotic Research, for which a browser emulation was necessary due to the use of a Flash animation which cannot be rendered by current web browsers. The artists had previously replaced dead links on the website with Internet Archive links (via migration). If Internet Archive links are followed in a browser emulation with an obsolete browser, the pages they lead to do not work, as they contain modern JavaScript. However, if the website is viewed without a browser emulation, the Flash does not work. Temporal consistencies between such measures therefore lead to dysfunctionalities, and must be straightened out (Espenschied and Moulds 2019, 435).

6.4 Long-Term Preservation and Short-Term Adaptations for Exhibitions

The long-term preservation of artworks always comes with curatorial decisions: the network of care decides which of the work-defining properties are to be preserved, and this results in a fixed set of such properties. These decisions are taken and frozen before exhibitions are planned. However, there is another moment at which curatorial decisions are

taken: when an artwork in storage needs to be installed/activated. This includes adapting it to the local socio-technical conditions in which it is to be shown, and the exhibition context. This section discusses the tension between long-term preservation and the activation of the artwork in a specific exhibition context.

Some time-based media artworks, such as *TraceNoizer*, are suitable for a wide range of presentation formats, whereas others have a rather narrow presentation format, such as *Horizons*. LAN have, for instance, presented *TraceNoizer* in such disparate contexts as on a stage projected on a big screen and at a kind of information desk, where people could create clones and receive printouts of their clone profile like business cards ([Lee n.d.a], see section 5.9). The work was also included in online exhibitions such as the *READ_ME_festival* in 2002, *The Right to RE-* in 2018, and *From the Early Web to Web 3.0* in 2022.

Opening *TraceNoizer* in a so-called browser emulation[65] is another option for presenting the artwork. Using a browser emulation will indicate the age of the artwork, if the emulator of an obsolete computer system is made visible in the browser window. This option was suggested but not executed for *TraceNoizer*, as the piece works quite well with modern web browsers. Browser emulations slow down access to the artwork, as they need to start up a computer emulation remotely. With newer technology, such as emulators based on JavaScript, browser emulations will hopefully become faster, leading to a more viable variant of encapsulation as a preservation strategy.

Although the various presentations of *TraceNoizer* yielded very different performances, the website itself has not been adapted for exhibition purposes.[66] Variability in these presentations came from several sources: exhibition design and subject, the use of specific hardware, paraphernalia, and wall-mounted prints.

For *Horizons*, short-term curatorial decisions are limited, as the work is made for a physical exhibition space. To show the work, adaptations to features of the exhibition space such as the lighting, the position of the tracking sensor, video projectors, and speakers are necessary. Concerning the software, the responsive area for the tracking sensor must be set. The quality of the video pattern behaviour is largely a matter of finding the right parameters, which are not absolute, as they partly depend on the space. Quality control through

65 In Figure 29 and Figure 30 (section 5.7), this is referred to as front end emulation. In section 6.3, it is called browser emulation. It is a specific type of the strategy of encapsulation.

66 The only exception here is *Right to RE-* exhibition in 2018, which presented a documentation of *TraceNoizer* as it was no longer online at that time.

recording the sensor input and video output can provide support here. The sound volume for the piece is quite high, which is why it needs its own space, separate from other artworks. As the hardware is not visible, *Horizons* appears fairly ageless, hence why curation options are rather limited to decisions regarding its installation in the space and position relative to other artworks.

For *Shan Shui*, an artwork that uses the same software and a very similar hardware setup to *Horizons*, Mul – given that he is the artist and because the Dortmunder U commissioned the work for a permanent installation in 2021 – made further-reaching decisions than was possible for *Horizons*. He adapted the video dimensions and resolution to the exhibition space, and used one modern projector with an aspect ratio of 16:9 instead of two adjacent projectors with an obsolete aspect ratio of 4:3 each. This suited the space better, and had the added advantage of requiring less maintenance. The Dortmunder U is not allowed to change the aspect ratio and resolution of *Shan Shui*: the video dimensions were renegotiable in the hands of the artist, but not once it was installed in the museum. Short-term adaptations to the space are thus limited to the position of the projector and the speakers, and the software setting of the reactive space.

For artworks that have been preserved via the documentation strategy, additional options exist to adapt them to an exhibition context. For through documentation, the relevant network of care creates narratives. This network must select the archival material with care, agree on it with the artist, and include different perspectives. Making the artwork's archive publicly accessible would make for greater transparency regarding the material selected from it. For a new artwork version based on different curatorial decisions, it would be wise to create a new branch in a version control system, such as is the case for the artwork *Chinese Gold* (2006-ongoing) by UBERMORGEN (Barok et al. 2019, 110).

Besides curatorial adaptations of the artwork to fit the exhibition context, changes in the artwork's socio-technical environment can cause further short-term and long-term adaptations. For instance, certain tools used as part of an artwork might be geographically dependent, such as communication platforms (such as WeChat, a Chinese app with limited or no access in certain parts of the world); censorship might apply in certain locations; national copyright laws might cause problems; the language of the artwork might have to be adapted to the language of the place where the work is to be exhibited; national technical standards might result in incompatibilities; local firewalls might filter out the artwork website or services used by the artwork; and so on.

To summarise, the subject and design of an exhibition can provide important contextual information so as to limit any adaptations that an artwork may require. A good introduction to the work can provide sufficient contextual information for understanding it and its potential dysfunctionalities. The presentation of the work is another variable that can be adapted: the work can be shown full-screen, in a window, or in an iframe; the screen resolution can also be changed, and the emulator can be made either visible or invisible. Besides these "soft" presentation options, the peripherals used to present the artwork also have an important role to play. Obsolete peripherals convey a different experience than new hardware. For obsolete technology such as CRT monitors, which have since become rare, "emulated"[67] technology (such as a ffmpeg CRT transform filter; see Rancour n.d.) might be an alternative. At the level of conservation strategy, curatorial capture allows one to capture more than necessary for a single narrative, meaning that the artwork can be shaped according to the exhibition context. Another option is presenting internet-based artworks with an obsolete web browser (as per the encapsulation strategy), instead of a current browser.

6.5 Conservation Strategies and Artwork Acquisition

The process of selecting a conservation strategy is not as straightforward as it is depicted in the 2020 "Decision-Making Model for Contemporary Art Conservation and Presentation" (Cologne Institute of Conservation Sciences / TH Köln 2019, revised 2021). An artwork's significant properties are not set in stone at the beginning of the conservation process. Generally, agreeing upon a conservation strategy is an iterative process, where the technical possibilities are balanced against sustainability criteria and the significant properties. The outcome of a conservation strategy is not always predictable, however, and measures need to be tested. This has implications for the acquisition process, at which stage the conservation options should be kept as open as possible. Another reason for keeping these options open at this point is that the conservator does not yet know the artwork well, and might be lacking important facts about it.

A significant decision when acquiring an artwork is whether to acquire the hardware along with the software. The Museum Boijmans Van Beuningen did not want to purchase the hardware for *Horizons* in 2017 due to a limited conservation budget, and because the

[67] Here it is more of a simulation/reconstruction than an emulation. Unlike emulation, a simulation or reconstruction is not bound to directly translate each machine or instruction set command.

hardware would quickly become obsolete. However, the hardware with the software installed serves as a reference for the work's behaviour and functionality when executing conservation strategies later. For artworks that are installed on a web server and accessed from an unspecified client, acquiring the hardware is not necessary provided the artwork is already online or is going to be installed on a web server immediately. For artworks that consist of obsolete software for which the hardware no longer exists, acquiring the hardware can be postponed until the artwork is to be preserved.

If the hardware is not acquired, the software and its environment must be acquired instead, and the computer architecture must be described. This can be done by acquiring disk images, software packages, or virtual machines/emulators with the artwork installed on them. The virtual machine then needs to be saved in a sustainable format such as OVF,[68] including standardised metadata about computer resources. Its functionality must be checked regularly, and its virtualisation environment maintained. Virtual machines and disk images that are run in emulators can be used as a documentation of the installation. As such, it can be convenient to have both disk image and single software components.

Another question relating to acquisition is whether a software executable is sufficient or whether the source code is required. Executables are hardware-dependent and require either the right hardware or an emulator to run successfully. If the software is to be migrated to a different hardware, source code is needed, and it provides a certain flexibility regarding hardware and operating system. Source code also makes it easier to reconstruct (that is, reprogram) the artwork in case the software is no longer maintained, and the hardware cannot be emulated.

The documentation of the artwork at the time of acquisition is crucial, too. It is worth spending some time on this point, as the documentation created at acquisition could later be used for the strategy of curatorial capture and come to represent the artwork. Particularly if the artwork is acquired without hardware, or has external dependencies, capturing it or requesting captures is paramount and should not be postponed, as the socio-technical environment changes quickly. When capturing the artwork, the question of its network boundaries and time boundaries also readily comes up and must be discussed by artists and curators.

[68] OVF (Open Virtualisation Format) is a standard "to create a platform and vendor-neutral format to bundle up virtual machines into one or more files that can be easily transported from one virtualisation platform to another" (Portnoy 2012, 48). It uses the file extensions OVF and OVA.

Artworks are often a few years old when acquired. The web environment of internet-based artworks is likely to have changed in this time, and functionality might have been lost, as was the case for *TraceNoizer*. This makes it more difficult to restore and/or reconstruct the artwork. Web archives such as the Internet Archive can be very useful in such cases. Gathering archival documents from different perspectives can further support artwork reconstruction.

6.6 Sustainable Conservation Strategies in an Institutional Context: The Technosystem of Conservation

Financial resources, an institution's conservation policies, its preservation infrastructure, and accessible knowledge within and outside the institution all influence preservation decisions and the outcome of conservation strategies considerably. They build a context of institutional conservation that can be described as a technosystem: conservation takes place in a socio-technical conservation environment. I have therefore included the financial resources and policies of institutional caretakers in my sustainability criteria for the conservation of software-based art. The following section discusses the relevance of institutional resources and policies for preservation decisions. Not only are preservation decisions guided by resources and policies, but preservation infrastructure more generally also reflects preservation resources and policies, and has a significant impact on conservation. As such, the final section highlights the role of preservation infrastructure in software-based art conservation.

Preservation Resources and Policies

Sustainability should always be seen in the context of the financial resources of an institution taking care of its collection of software-based artworks. The criteria for sustainable preservation as defined in my theoretical framework (section 3.1) state that preservation strategies should lower or at least not increase the preservation effort needed for the following preservation cycle. Consequently, the sustainable preservation of software-based art requires an institutional conservation budget and policy that enable the perpetuation of the artwork in the long term.

In conservation ethics, financial considerations are hardly addressed, but are often crucial in the decision-making process. Only since 2020 and the new "Decision-Making Model for Contemporary Art Conservation and Presentation", has the question of financial

resources as a decisive factor come to the fore (Cologne Institute of Conservation Sciences / TH Köln 2019, revised 2021, 14). The sustainability and technosystem theory acknowledge this important fact of conservation practice, as sustainable preservation is not possible without long-term financial support.

The functioning of *TraceNoizer*'s clone engine is a significant property. However, having updated the search engine API, the quality of the clones was not satisfying. I was ready to compromise on the quality of the clones produced, as I was afraid that the House of Electronic Arts would spend too much money on analysing the problem without being able to improve the outcome significantly. A large institution might have a larger conservation budget per artwork. If resources are scarce, one option is to compromise on the significant properties of the artwork in return for less effort being required for long-term maintenance or conservation. If an institution applies this to a significant number of its artworks, it should mention it in its conservation policy. Rhizome is an example of an institution that reduces the effort needed to maintain and conserve its artworks by accepting a certain degree of damage caused by ageing, applying the strategies of documentation, reconstruction, and encapsulation, and scaling the encapsulation strategy by using a common virtualisation environment. This means that Rhizome has adapted their preservation policy so that the presentation of software-based art does not necessarily correspond to the function of Display as theorised by Keene (see section 2.5 and footnote[69]), but rather sits somewhere between Evidence and Display. It also indirectly demonstrates the role of financial resources in the development of feasible conservation policies.

Another example of a conservation policy is the fact that the House of Electronic Arts tries to buy as little hardware as possible. This is due to the small amount of space that they have for physical artwork storage, itself also a consequence of a limited conservation budget. As such, the equipment they use for exhibitions either comes from their hardware pool, or they acquire virtual machines. There is a risk to this policy, namely that there are no working reference installations, either because the virtual machines cannot be opened or are not configured properly. It does, however, force conservators to think about the process of transferring artworks to new hardware as early as the moment of acquisition, and to request virtual machines that contain the necessary software environment instead of computer installations, as well as to request the right documentation.

[69] According to (Keene 2002, 17), there are three purposes for a museum collection: aesthetic experience (Keene calls it display in her triangle of purposes), typical of art museums, evidence, typical of archaeological museums, and demonstration, typical of science museums.

To conclude, conservation policies are crucial for conservation decisions, and are only sustainable if they take into consideration conservation resources.

Preservation Infrastructure

Preservation infrastructure comes as a consequence of an institution's preservation policy. As in painting or sculpture conservation, infrastructure such as conservation labs, collection storage, exhibition spaces, and loading bays are necessary for the professional handling of artworks. Drawing on my experience as a conservator at the House of Electronic Arts, at LI-MA, and from my case studies, this part of my analysis looks at how a relevant piece of infrastructure such as a media lab, digital archive, digital exhibition, or documentation infrastructure (see Figure 39) may support the sustainable conservation of software-based art. Such infrastructure is not yet self-evident in an art museum, which will be illustrated below.

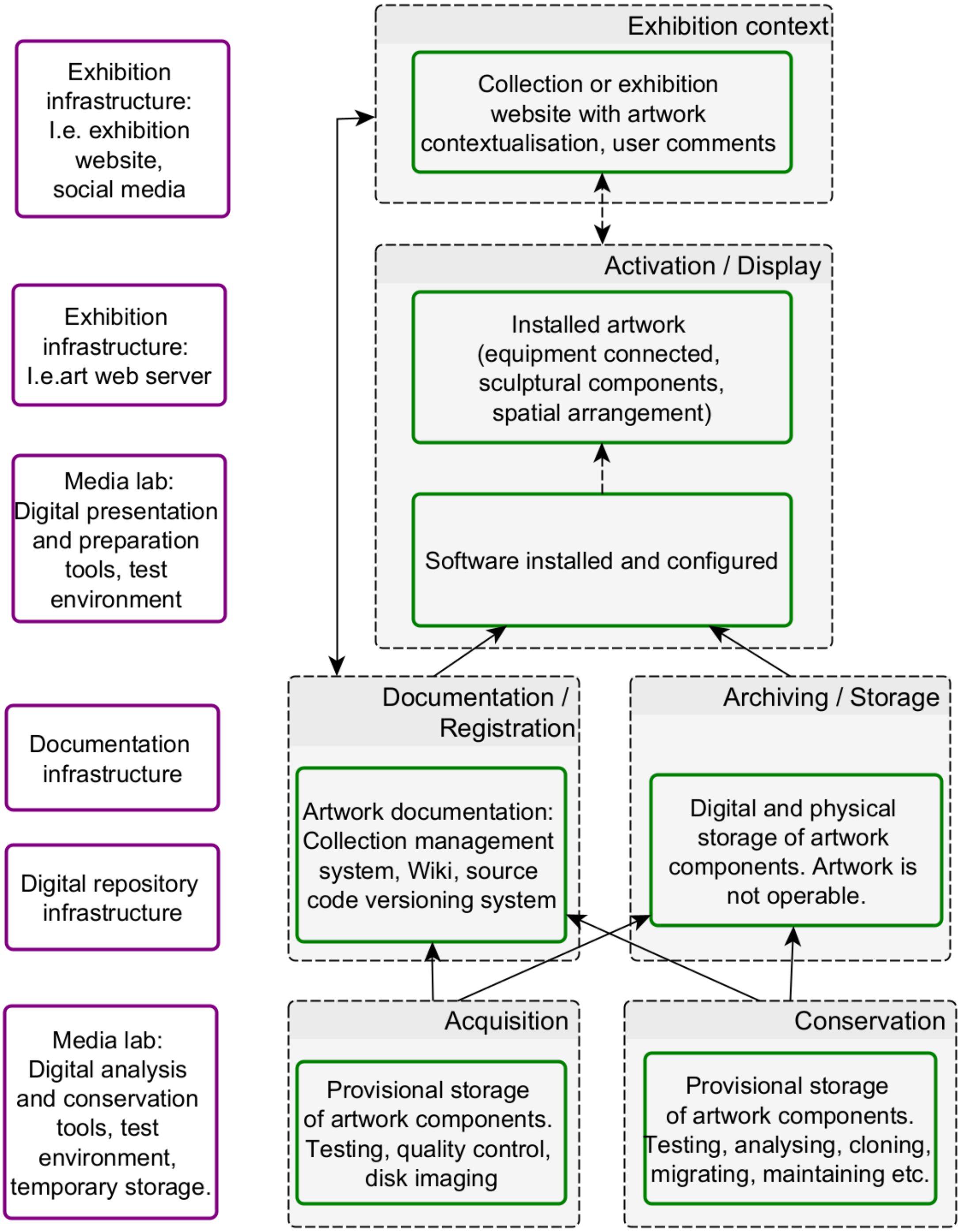


Figure 39: Contributions of various different infrastructural building blocks to the preservation of software-based artwork. The arrows symbolise the transfer of digital data from one process to the next.

A good media lab generally improves the acquisition process and preservation quality, as it gives the conservator access to the artwork, an opportunity to analyse and check the artwork, and the possibility to secure software and data. For instance, for *Horizons*, a disk image of the artist’s computer and copies of the source code were made at

LI-MA's media lab, as the Museum Boijmans Van Beuningen did not have such a lab. This meant that tests could safely be carried out on the disk image and the copied source code without endangering the artwork. A media lab also contains devices with which to read obsolete media carriers and connect obsolete peripherals and sensors, provides different operating systems and installations of emulators and/or virtual machines for testing and preservation, and supports the acquisition, display preparation and preservation of software-based art. Outsourcing the media lab would require the institution to also outsource the acquisition process, and would mean that they were not able to carry out conservation strategies. However, a media lab can be simplified by outsourcing certain tasks, such as exporting data from obsolete media carriers.

Infrastructure for Emulation-as-a-Service – such as EaaSI – as has been implemented in university libraries and archives in North America, can prove a positive addition to a media lab (see section 2.4). Here, Annet Dekker's concept of a network of care converges with a technological network of institutions that share software environments for emulations. However, apart from Rhizome, I am unaware of any art institution that serves as a node in the Emulation-as-a-Service infrastructure. Although in need of this service (for browser emulations, for instance), HEK has not yet had sufficient resources to run its own node, and was helped in this respect by Rhizome. Emulation-as-a-Service is especially useful for institutions that hold digital artefacts without hardware such as *TraceNoizer* and other works of internet-based art. For software-based artworks that often depend on specific hardware (including the connection of peripherals), require good performance at exhibitions, and whose software artefacts are non-standard, Emulation-as-a-Service has fewer advantages. Moreover, copyright laws still limit the sharing of software environments in Europe. However, Emulation-as-a-Service is suitable for making certain software-based artworks accessible online and is useful for preparing locally installed emulations, as well as for testing and analysing artworks. It does not lead to additional conservation risks, as the emulators used by EaaSI are available independently from the EaaSI platform.

Another important building block of digital infrastructure is the digital repository (that is, the digital archive) that corresponds to a museum collection's physical storage. The storage strategy relies on the existence of such a digital repository. Best practice recommendations for digital archives, such as those recommended by (The National Digital Stewardship Alliance 2019) or on the Matters in Media Art website (MoMA et al. n.d.), as well as the ISO standard for so-called trustworthy repositories that conform to higher, more

professional standards (ISO/IEC 2012),[70] are designed to make repositories more resilient against bit rot, human errors, and other risks. A digital repository can be shared or outsourced, as the archival processes can be automated and professionalised when there is a larger number of digital objects. LI-MA's digital storage is an example of shared infrastructure, provided as a service by LI-MA to custodians of media art collections in the Netherlands[71] such as the Museum Boijmans Van Beuningen.

A digital exhibition infrastructure may consist of an art web server through which single artworks are made accessible online. An art web server needs to be able to provide artwork-specific – and often obsolete – web server environments. Rhizome, the House of Electronic Arts, and LI-MA currently run such servers. LI-MA offers it as a service to collections in the Netherlands, as commercial hosts regularly force artists and museums to migrate their internet-based artworks over to newer software environments. Digital exhibition infrastructure may also consist of an exhibition platform where online artworks can be presented, similar to a physical space where artworks are curated according to the exhibition topic[72]. The latter is an example of the short-term curation of artworks (see section 6.4), as a descriptive or explanatory text can be added to each piece as well as the exhibition as a whole.

Documentation infrastructure should facilitate processes of collaborative conservation, whilst also supporting change management and preservation planning for software-based artworks. For instance, the source code of *Shan Shui* was managed in a multi-user version control system and the source code changes were explained in a Wiki (Roeck and Gaure 2021a). Artist, programmer, and conservator can all access and modify the source code and its documentation. In theory, preservation tasks could also be managed in the versioning control system by creating so-called issues. Whether or not this is a practical solution would have to be tested.

To summarise, without digital infrastructure and its related processes, efficient and reliable long-term preservation is simply impossible. Theoretically, all the conservation infrastructure can be outsourced. However, a digital media lab with tools for artwork analysis and conservation as well as infrastructure for documentation allows institutions

---

70 For instance, for repositories that offer their services to external clients.

71 Information about LI-MA's storage service see their website (LI-MA n.d.b) and navigate to preservation and then to storage.

72 For instance, the digital exhibition platform common garden designed by Constant Dullaart and implemented at HEK as *virtual.hek.ch*.

more control over their processes of acquiring and conserving software-based artworks. As the sociologist Susan Leigh Star puts it, infrastructure is interlinked with museum practice (Leigh Star 1999, 381–82). Besides, conservation infrastructure is part of the technosystem of an artwork and therefore influences its preservation.

7 Conclusion: How to Sustain Software-Based Art

As contemporary artworks, software-based artworks have a certain variability; sometimes their boundaries are unclear, but they always have an environment. Whilst it is this fast-changing socio-technical environment that is often the cause of dysfunctionalities in the immediate term, these characteristics also represent a challenge for preservation in the long term. Furthermore, and as past examples have shown, conservation strategies such as migration can create new problems in the near future, if they are not carefully chosen and executed with sustainability criteria in mind. Finally, it is not unusual to find that, when an artwork in an institutional collection has to be restored, crucial components such as the computer or software are no longer traceable, or that necessary documentation is missing.

This dissertation provides a theoretical framework with which to tackle these challenges for long-term preservation in an institutional context. Following my experience of the case studies in chapters 4 and 5, I broke the framework down into more concrete approaches and requirements to sustain software-based art. The framework also adjusts the definitions of certain conservation strategies in terms of the technosystem, and explores ways in which they may be useful for long-term preservation. My goal has been to present an approach to identifying conservation strategies for software-based artworks, and to explore the institutional prerequisites for sustaining such art.

The theoretical framework unites the questions of digital materiality, the socio-technical system (technosystem), and sustainability criteria for long-term preservation. I have identified the digital materiality of an artwork as an important factor in choosing preservation strategies. Software characterises software-based artworks as clay does in the case of sculptures; through its limitations and idiosyncrasies, software influences the artwork's behaviour whilst also implementing its functionality. Another aspect of digital materiality is that it is an expression of the interaction between software and hardware, also articulated in the software performance model examined in section 2.2 (A. Wilson 2005). This approach runs counter to the supposed ephemerality of software. Lastly, software also has a socio-cultural and historical character, which adds to its meaning as both a tool and as digital material. Riegl and Appelbaum's values of age value, aesthetic value, historical value, artistic value, and use value can be taken to characterise and identify a software-based artwork's digital materiality (Appelbaum 2007, 89; Riegl 1982). Neither Appelbaum nor

Riegl mentions socio-cultural value; however, I contend that for a comprehensive assessment of an artwork's materiality, its socio-cultural value must be considered. I further suggest using a grid – with the artwork's components and layers as rows and the associated values as columns – to implement the description of digital materiality in the conservation of software-based art (see Appendix A.1). A knowledge of digital materiality is likely to produce conservation strategies that stay closer to the material, and this corresponds to the minimal intervention principle of conservation ethics.

Acquiring the knowledge necessary to assess an artwork's digital materiality on these various levels is not traditionally a part of conservation education and technical art history. However, software engineers and computer scientists themselves often lack an awareness of digital heritage or a respect for the possible historical and cultural significance of obsolete code. This situation therefore requires collaboration between professionals across different fields, as well as interdisciplinary learning and education.

The second part of my theoretical framework, heavily informed by Feenberg's concept of technosystem, acknowledges that a software-based artwork is a technical system situated in a socio-technical environment. First and foremost, Feenberg's technosystem implies an approach to analysing artworks as systems of building blocks and layers, and this facilitates the identification of suitable conservation strategies. The complexity of digital artworks can be overwhelming, which is why reducing complexity on a conceptual level (by breaking them down into constituent building blocks and layers) can lead to resolving conservation problems. A building block is any part of an artwork that can be treated with its own conservation strategy independently from the rest of the artwork. Such blocks can exist next to each other and do not have to be hierarchical. By contrast, a functional layer is a horizontal cross-section of any hierarchical software stack. It can be the point of departure for a conservation strategy such as encapsulation or bridging. The artwork's environment is the third conceptual element that I consider paramount for the conservation of software-based artwork. As mentioned in section 6.1, software-based artworks are fragile due to their dependencies on the outside world. If their environment is taken into consideration as part of their conservation, their chances for long-term preservation increase. The technosystem approach worked successfully for the software-based artworks *TraceNoizer* and *Horizons*, as proved in section 6.2.

The sustainability criteria for conservation strategies constitute the third part of my theoretical framework. As we know from past experience, conservation decisions are often

not sustainable[73]. One obsolete technology is replaced by another technology which becomes obsolete a few years later, partly rendering the conservation effort futile. The aim of the sustainability criteria is to decrease the risk of having to repeat costly conservation measures. This can be achieved by reducing the structural complexity of the artwork in question, such as its external dependencies, and by stabilising source code complexity (as opposed to increasing it). Dependencies can be reduced via both reconstruction and documentation, by replacing them with either a reconstructed part or a captured output. In contrast to encapsulation, migration increases source code complexity and hence the maintenance needed by the artwork in the long term. Conservation strategies that are scalable, such as encapsulation, reduce the maintenance per artwork and the risk of introducing errors due to the use of a common infrastructure. However, the scaling of encapsulations is viable for web-based artworks that have similar, unspecific hardware requirements; for other software-based artworks it is much more difficult to realise. Moreover, conservation strategies should not impede future generations from preserving artworks with new technologies and knowledge.

Unsurprisingly, applying these criteria to the *TraceNoizer* and *Horizons* case studies proved that compromises must be accepted between what would theoretically be the most sustainable conservation strategy and the artwork's work-defining properties. This can in turn lead to conservation strategies that do not necessarily excel in terms of sustainability (see section 6.2). However, the passing of time can also benefit certain conservation strategies, such as emulation, as computer processing power increases quickly and emulator technology improves, thus overcoming certain limitations in the future.

To ensure the sustainability of conservation strategies, the network of care has duties too. It must stabilise an artwork's significant properties to a certain extent, otherwise conservation strategies will have to be revisited again and again, and conservation costs cannot be reduced. At first glance, this goes against the paradigm of the variability of media art. However, as discussed in section 6.4, long-term conservation has options for short-term curation that can encompass certain variable properties of the artwork. This includes framing the artwork in given ways through exhibition design, labels, curatorial texts, or presenting it through specific hardware or spatial arrangements. It may also involve modifying the visibility of the browser or emulator window, or adapting the selection of

---

[73] One such instance involved *TV Bot* (2004, 2010) by Marc Lee (Gassert 2013). There is now a third version of *TV Bot* (2016), see (Lee 2004c).

captured or archival material to the exhibition context in the case of the documentation strategy.

One way in which this dissertation has contributed to academic research on the conservation of software-based art is by harmonising the definitions of certain conservation strategies in the fields of digital preservation and contemporary art conservation. As software-based art is situated at the cross-section of these fields, the terms used in relation to it should be unambiguous. In section 6.3, I suggested renaming the emulation strategy of the Variable Media Network "reconstruction", to avoid confusion with the strategy of digital emulation from the field of digital preservation. Through reconstruction, the goal is to achieve the same look and/or function of the original art object, without necessarily using the same (software) material or structure. I also argued in favour of referring to digital emulation as "encapsulation", so that the term encapsulation can be used outside of digital technology in fields such as contemporary art or for new technologies. A definition of conservation strategies that is independent of technology enables discussion and comparisons of them in other conservation fields. Encapsulation protects the artefact from unintended changes by encapsulating it within a layer, so as to be able to instantiate (activate) the artwork. I include bridging within the set of conservation strategies as a sub-strategy of encapsulation. Bridging adds a local patch between the environment and the artwork to bridge between them, so that changes in the environment can be intercepted. Bridging is thus a partial encapsulation. Besides, I support Dekker and Giannacchi's claim that documentation is a fully-fledged conservation strategy (Dekker and Giannachi 2022, 4), although it has been rarely applied to software-based art in contemporary art museums until now. The documentation strategy is appropriate for artworks that cannot be migrated, encapsulated, or reconstructed, and for which the only other feasible strategy is reinterpretation. Indeed, documentation and reinterpretation complement each other, as an artwork can only be reinterpreted if there are memories and or documentations of it. Migration and encapsulation are not possible for artworks based on external dependencies such as social media platforms or requiring a specific piece of hardware on which they rely for functionality, and which also have specific meanings associated with them. Whilst museum conservators are hesitant to explore this approach, the example of *TraceNoizer*, where LAN archived clones and made them accessible, shows that artists are less averse to such an approach (see section 6.3).

Often, documentation is combined with other strategies. One such typical combination is the reconstruction strategy interwoven with captures of the artwork to

replace external dependencies and thus reduce structural complexity. This combination is also useful for stabilising the artwork's environment. Other types of combination are also possible, such as nesting a migration within an encapsulation, as shown for *Horizons* (section 4.6), or linking encapsulations in a virtual network, such as in the emulation version of *TraceNoizer* (section 5.7). Furthermore, the subdivision of an artwork into building blocks allows for different conservation strategies in parallel, an example of which was the migration version of *TraceNoizer* (section 5.6). Although such combinations must be common, I argue that this analysis of combined conservation strategies is new and supports conservators in their efforts to identify sustainable conservation solutions.

In line with existing scholarship (such as Ensom 2018, 73]), I argue that the acquisition process of a software-based artwork is crucial to its long-term preservation. It is at this moment that the artwork is most likely to still be functional, and the artist is most likely to remember the production process. As the artwork's socio-technical environment changes quickly, however, not acquiring its hardware represents a considerable risk, as it may become unavailable, and cannot be used as a reference for the functionality and behaviour of the work. Therefore, if the hardware is not acquired, a description of the hardware architecture and performance, as well as the software environment, will have to be procured. Whether an executable of the software is sufficient, or whether the source code also has to be acquired, depends on the conservation strategy. Source code is needed when the software has to be migrated to a different piece of hardware, and provides a certain flexibility regarding hardware and operating system. Source code also makes it easier to reconstruct (reprogram) the artwork in case the software is no longer maintained, and the hardware cannot be emulated.

To enable the long-term preservation of software-based art, the theoretical framework must consider the artwork's institutional context. Whereas Saaze et al. studied how MoMA adapted to the requirements of time-based media conservation (Saaze, Wharton, and Reisman 2018), my research asks what is required to conserve software-based art sustainably. The institutional context – its financial resources, policies, infrastructure, and knowledge – has a significant impact on the preservation of its collection and influences conservation results. This is why the theoretical framework states, in section 3.1, that conservation strategies are only sustainable if the institution is able to provide a conservation budget and policies that enable the perpetuation of the artwork in the long term. This requirement is derived from the field of digital preservation. Such a budget does

not only include the direct conservation of artworks, but must also account for the infrastructure necessary for preservation and maintenance.

If financial resources are lacking, museums may start to prioritise the treatment of certain artworks. This dissertation does not provide criteria to select which artworks to preserve, as – according to the ICOM Code of Ethics for Museums – the museum is responsible for the stewardship of all of its artworks (that are not designated for deaccession). Another option, if resources are scarce, is to compromise on an artwork's significant properties so as to reduce the efforts required for long-term maintenance and conservation. This can result in accepting a certain level of damage, either by doing nothing, or by implementing a documentation strategy (by using captured or archival material, for example) and/or a reconstruction strategy (by replacing external dependencies). Backward changes through loss of functionality, which may involve presenting the artwork in a dilapidated state, or forward changes through migration and updating, are two ways of preserving digital heritage, if no better options are available. Backward changes usually need contextualisation to explain the artist's intention and the reason for these deficits.

A digital preservation infrastructure is a prerequisite to be able to carry out conservation. As Leigh Star contends, infrastructure is a part of human organisation (Leigh Star 1999, 380), enabling and shaping conservation processes. And yet, infrastructure or the associated tools are rarely discussed in conservation research. In painting and sculpture conservation, infrastructural elements such as a workspace and storage are taken for granted. For software-based art (and other time-based media art such as video art), however, this is not yet the case. Museums that do not dispose of a media lab themselves must outsource conservation support for both the acquisition and conservation of software-based art. By contrast, the digital archive necessary for the strategy of storage is an infrastructure that can be outsourced without hindering the acquisition process or the execution of conservation strategies. A web server for internet-based art is another conservation infrastructure that can be shared or outsourced, even if only to specialised hosts (most commercial hosts are unable to provide this service). Outsourcing such infrastructure nevertheless still requires a basic understanding of digital preservation processes and conservation strategies, otherwise the wrong choices may be made and conservation compromised.

It is therefore important for museums to have a policy regarding the preservation of software-based art, defining what they do themselves, what they outsource, how they deal with the hardware components of such art, what conservation infrastructure they run, and how they share knowledge about software-based art conservation. Given that many art

museums already have conservation policies, it would simply be a matter of adding software-based art to these, and making their policies publicly accessible to show their commitment to the conservation of their collection. Policies are the link between theory and practice, and are especially important for institutions with a public mission. Policies for the conservation of software-based art support the network of care by making sustainable conservation choices and creating the institutional environment necessary to accomplish sustainable conservation strategies.

In the technosystem, Feenberg describes the danger of technologies getting out of control and negatively influencing human communities and nature, thereby creating a dystopian future as a consequence of modernity. The solution, as he sees it, resides in "effective communication between lay and expert actors—between, in other words, public protest and technical implementation" (Feenberg 2017, 114). This ties in with Annet Dekker's suggestion that artworks should be cared for through a network of professionals and non-experts (Dekker 2014, 81), as opposed to relying solely on institutions. This would push institutions to seek more collaboration with communities outside of themselves, and to commit to change not merely superficially, but at the level of their structure, goals, and activities. This would also enable them to tell more multifaceted stories through their collection. Kemp shares this idea, stating that the model of the distributed version control system could be applied to conservation in a figurative manner, making visible the contributions of diverse groups to an artwork (Kemp 2022, 9/16).

Recommendations for Future Research and Final Thoughts

Based on the results of the research conducted for this dissertation, I will suggest a number of topics for additional research that will further contribute to the sustainable conservation of software-based art.

One of the underlying assumptions of this dissertation is that software is based on digital technology. Although I defined the conservation strategies discussed here as being independent from technology, both the framework and strategies would have to be tested for compatibility with other technologies such as quantum computing (if this becomes relevant for artistic creation). The same goes for the application of the strategies to contemporary art conservation.

My research methodology comprises a comparison of two conservation strategies applied to a piece of software-based artwork. Long-term experimentation was not possible within the period of research for this dissertation. Observing the conservation of software-

based artworks over a longer period – say, at least twenty-five years – whilst applying the sustainability criteria would yield a great deal of insight into the criteria's practicability. Also of interest would be to study how an artwork develops over such a long period if the same strategy is applied repeatedly or prolonged over time. If this was conducted with two different strategies, such as emulation and migration, the resulting versions of the artwork could be compared after a given period and conclusions drawn about the change of the artwork, as well as about the maintenance and conservation effort involved.

It is difficult to generalise and recommend certain strategies for specific types of artwork due to the fact that identifying an artwork's significant properties is highly subjective. However, some conservation strategies are applied more frequently to certain types of artwork than others. I compared contained software-based artworks with internet-based artworks in section 5.8. Elsewhere, the time-based media conservators Tom Ensom and Jack McConchie have created a collaborative website that contains practical recommendations for the preservation of virtual reality-based artworks (Ensom and McConchie since 2021) that are highly dependent on peripherals such as virtual reality headsets and sensors. It collects information about VR-specific file formats and game engines along with any necessary considerations for migration to newer hardware. Other software-based artwork genres to be investigated include AI-based or robot-like works whose behaviour is controlled through machine learning, and software-based artworks that are integrated in a blockchain (Web 3.0).

Following this, research potential also exists in looking more closely at sustainable migration practices, which is to say, creating reproducible and well-documented migrations that last as long as possible, by for example updating to a long-term support version or by migrating to an open-source format. Avoiding hacker solutions that exploit software weaknesses could be part of such practical advice, as software companies are likely to repair this kind of weakness and the solution will no longer work.

The influence of emulation developers on the strategy of encapsulation should not be underestimated. Encapsulation's potential for conservation is theoretically high in terms of its technical feasibility. However, some computer hardware such as sound and graphics cards are not sufficiently commercially interesting for good emulations to be provided for them. For some emulators, improvements to user-friendliness would enable more conservators to use them (not having to recompile an emulator to integrate a virtual video card, for instance, would facilitate its application). Preservation features such as regulating the clock speed of the emulated CPU, or the possibility to measure and compare video and

audio output, would make encapsulation much easier to implement. For web-based artworks, faster browser emulations (those based on JavaScript, for example) would be helpful for running old web browsers. Such conservation-specific improvements would, of course, require funds for software development as well as for institutions to focus on certain emulators. However, this could be a typical subject for a conservation policy to increase its chances of success.

Finally, I would like to expand on some assumptions that undergird this dissertation. As Latour stated in 1993, modernism is accompanied by historicism, and "Maniacal destruction is counterbalanced by an equally maniacal conservation" (Latour 1991/1993, 69). Western society's maniacal obsession with new products and the concurrent cycles of obsolescence and oblivion that characterise our relationship to digital technology have informed my personal preference for minimal intervention, as well as an appreciation of the materiality of software-based artworks as witnesses to the past. Growing up in a wealthy town, I witnessed how money can lead to the destruction and over-renovation of many historical buildings, often hollowing them out to leave only the skin. As such, poorer communities often end up having better preserved built heritage, despite having had less money to spend on it. Given the little attention paid to the preservation of digital heritage, even though digital technology continues to develop extremely fast, I think that there is good reason to be conservative as a conservator, and prioritise the preservation of materiality over functionality more often. It could be argued that built heritage differs from software-based art, as built heritage has a use value (to recall Riegl's list of values) whereas art only has artistic value. Yet both built heritage and software-based art are socio-technical practices and both face dysfunctionalities due to changing patterns of use or environment. The network of care can make an important contribution to digital heritage if it can accept not only forward changes in the form of migration and updates, but also backward changes such as the loss of functionality. In other words, even though prioritising the material over functionality can lead to ruins, these may still represent the artefact and the artist's intention, despite dysfunctionalities.

The concept of the "great acceleration" describes the growing speed of both innovation and consumption within Western society which now defines our current geological era, the Anthropocene (Head et al. 2021, 359). The short lifespan of software, and therefore of software-based art, could be seen to indicate this great acceleration. The job of the conservator is to try to slow down this acceleration for a minority of the software-based artworks that are in institutional collections. However, due to their being embedded in

the technosystem, this is a challenge. A general deceleration of the entire technosystem – through improvements to the quality of software and hardware, abandoning planned obsolescence, and ramping up the capacity of repair services – would of course be highly beneficial, but exceeds the abilities of the network of care. Such measures would instead require political and social change as suggested by Feenberg. In the meantime, the network of care can draw on the analyses presented here as a means of ensuring the best conditions for carrying software-based artworks towards the future.

References

Acker, Amelia. 2020. *Emulation Practices in Place: Coordinating Software Preservation in Libraries, Archives, and Museums: Report from 2019 Summer Fieldwork.* University of Texas. Accessed November 13, 2023. https://www.softwarepreservationnetwork.org/wp-content/uploads/2020/06/Emulation-Practices-in-Place-Report_Acker_RESEARCH.pdf.

Amoretti, Francesco, and Mauro Santaniello. 2009. "Community of Production." In Pagani 2009, 224-229.

Appelbaum, Barbara. 2007. *Conservation Treatment Methodology.* Oxford: Butterworth-Heinemann.

Archibald, Jake, and Marijn Kruisselbrink. 2022. "Service Workers: W3C Candidate Recommendation Draft 12 July 2022." Accessed November 13, 2023. https://www.w3.org/TR/service-workers/.

Barad, Miryam. 2018. *Strategies and Techniques for Quality and Flexibility // Strategies and Techniques for Quality and Flexibility.* SpringerBriefs in applied sciences and technology, 2191-530X. Cham, Switzerland: Springer.

Barok, Dušan, and Julie Boschat Thorez. 2020. "Naked on Pluto/Documentation." Accessed November 30, 2023. https://monoskop.org/Naked_on_Pluto/Documentation.

Barok, Dušan, Julie Boschat Thorez, Annet Dekker, David Gauthier, and Claudia Roeck. 2019. "Archiving Complex Digital Artworks." *Journal of the Institute of Conservation* 42 (2): 94–113. doi:10.1080/19455224.2019.1604398.

Barok, Dušan, Julia Noordegraaf, and Arjen P. de Vries. 2019. "From Collection Management to Content Management in Art Documentation: The Conservator as an Editor." *Studies in Conservation* 64 (8). https://www.tandfonline.com/doi/full/10.1080/00393630.2019.1603921.

Barrus, John. 2016. "Announcing GPUs for Google Cloud Platform." Accessed December 11, 2023. https://cloud.google.com/blog/products/gcp/announcing-gpus-for-google-cloud-platform/.

Baumgärtel, Tilman. 2001. *[Net.Art 2.0]: Neue Materialen Zur Netzkunst / New Materials Towards Net Art.* Nürnberg: Verlag für moderne Kunst.

Becker, Christoph, Hannes Kulovits, Mark Guttenbrunner, Stephan Strodl, Andreas Rauber, and Hans Hofman. 2009. "Systematic Planning for Digital Preservation: Evaluating Potential Strategies and Building Preservation Plans.".

Belady, Laszlo A., and M.M. Lehman. 1985. *Program Evolution : Processes of Software Change.* London: London : Academic Press.

Berkeley Art Museum / Pacific Film Archives Berkeley, Franklin Furnace New York, Guggenheim Museum, Daniel Langlois Foundation for Art, Science and Technology Montréal, Performance Art Festival + Archives Cleveland, Rhizome, and Walker Art Center Minneapolis. 2001. "The Variable Media Network Website." Accessed November 25, 2023. https://www.variablemedia.net/e/welcome.html.

Bettivia, Rhiannon. 2016a. “Mapping Significance of Video Games in OAIS.” In ipres 2016 2016, 212–20.

———. 2016b. “Where Does Significance Lie: Locating the Significant Properties of Video Games in Preserving Virtual Worlds II Data: Enrolling Heterogeneous Partners in Video Game Preservation.” *International Journal of Digital Curation* 11 (1): 17–32. doi:10.2218/ijdc.v11i1.339.

Bijker, Wiebe E. 1995. *Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change / Wiebe E. Bijker.* Inside technology. Cambridge, Mass., London: MIT Press.

Bosshard, Andres. 2020. “Telefonia-2021-1991-1291: Online Archive of the Live Event.” Accessed November 30, 2023. https://telefonia.hek.ch/.

Bowers, Jason, John May, Erik Melander, Matthew Baarman, and Azeem Ayoob. 2002. “Tailoring XP for Large System Mission Critical: Software Development.” In *Extreme Programming and Agile Methods: XP/Agile Universe 2002 : Second XP Universe and First Agile Universe Conference, Chicago, IL, USA, August 4-7, 2002 : Proceedings / Don Wells, Laurie Williams (Eds.)*, edited by Don Wells and Laurie Williams, 100–111. Lecture Notes in Computer Science 2418. Berlin, London: Springer.

Brain, Tega. 2018. “The Environment Is Not a System.” *APRJA* 7 (1). https://aprja.net//article/view/116062/164244. Accessed November 13, 2023.

Bundesamt für Statistik Schweiz. n.d. “Internetzugang Der Haushalte: Tabelle. 2002-2021.” BFS-number 30109. Accessed December 01, 2023. https://www.bfs.admin.ch/bfs/de/home/statistiken/kultur-medien-informationsgesellschaft-sport/informationsgesellschaft/gesamtindikatoren/haushalte-bevoelkerung/internetzugang-haushalte.assetdetail.25065502.html.

Burke, Verity, Dolly Jørgensen, and Finn A. Jørgensen. 2020. “Museums at Home: Digital Initiatives in Response to COVID-19.” *Norsk museumstidsskrift* 6 (2): 117–23. doi:10.18261/issn.2464-2525-2020-02-05.

Butterfield, A., Anne Kerr, and Gerard E. Ngondi. 2016. *A Dictionary of Computer Science.* Seventh edition. Oxford quick reference. Oxford: Oxford University Press.

Caianiello, Tiziana. 2013. “Materializing the Ephemeral: The Preservation and Presentation of Media Art Installations.” In *Medienkunst-Installationen: Erhaltung Und Präsentation ; Konkretionen Des Flüchtigen = Media Art Installations ; Preservation and Presentation ; Materializing the Ephemeral*, edited by Renate Buschmann and Tiziana Caianiello. Berlin: Reimer.

Castriota, Brian. 2019a. “Authenticity, Identity, and Essentialism: Reframing Conservation Practice.” In *What Is the Essence of Conservation? Materials for a Discussion*, edited by François Mairesse and Renata F. Peters.

———. 2019b. “Securing a Futurity: Artwork Identity and Authenticity in the Conservation of Contemporary Art.” doctoral thesis, School of Culture and Creative Arts, University of Glasgow.

Cheang, Shu Lea. since 1998. “Brandon (1998-1999).” Accessed November 25, 2023. http://brandon.guggenheim.org/.

Cologne Institute of Conservation Sciences / TH Köln. 2019, revised 2021. *The Decision-Making Model for Contemporary Art Conservation and Presentation.* Cologne (D). Accessed November 26, 2023. https://www.th-koeln.de/en/decision-making-model-for-contemporary-art-conservation-and-presentation--revisited_63959.php.

Connor, Michael, Aria Dean, Michael J. Connor, and Dragan Espenschied, eds. 2019. *The Art Happens Here: Net Art Anthology.* New York: RHIZOME.

Cooper, I., I. Melve, and G. Tomlinson. 2001. “Internet Web Replication and Caching Taxonomy: Request for Comments: 3040.” Accessed December 01, 2023. https://www.rfc-editor.org/info/rfc3040.

Davoli, Renzo. 2005. “VDE: Virtual Distributed Ethernet.” In *International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, 213–20: IEEE Computer Society.

Dekker, Annet. 2014. “Enabling the Future, or How to Survive Forever: A Study of Networks, Processes and Ambiguity in Net Art and the Need for an Expanded Practice of Conservation.” Doctoral Thesis, Centre for Cultural Studies, Goldsmiths, University of London. Accessed October 19, 2023. https://research.gold.ac.uk/id/eprint/11155/.

Dekker, Annet, and Gabriella Giannachi, eds. 2022. *Documentation as Art: Expanded Digital Practices.* 1st. London: Routledge.

———. 2022. “Introduction.” In Dekker and Giannachi 2022, 3–15.

Delve, Janet, Leo Konstantelos, and Antonio Ciuffreda. 2011. “TOTEM: Trusted Online Technical Environment Metadata - a Long-Term Solution for a Relational Database / RDF Ontologies.” In *IPres 2011: 8th International Conference on Preservation of Digital Objects*, edited by iPres 2011.

Derrida, Jacquues. 1967/1988 (reprint). “Structure, Sign, and Play in the Discourse of the Human Sciences.” *Modern criticism and theory*, 107–23.

Digital Preservation Coalition. 2015. “Digital Preservation Handbook.” Accessed November 13, 2023. https://www.dpconline.org/handbook.

DOCAM. 2010a. “2.1 Conservation Strategies: 2.1.1 Emulation.” In *Conservation Guide*. Conservation Guide: DOCAM. Accessed November 25, 2023. http://www.docam.ca/en/21-conservation-strategies/21-emulation.html.

———. 2010b. “2.1 Conservation Strategies: 2.1.6 Reconstruction.” In *Conservation Guide*. Conservation Guide: DOCAM. Accessed November 13, 2023. https://www.docam.ca/en/21-conservation-strategies/216-reconstruction.html.

———. 2010c. “A Decision-Making Model: The Decision Tree.” Accessed November 13, 2023. http://www.docam.ca/en/restoration-decisions/a-decision-making-model-the-decision-tree.html.

———. 2010. *Conservation Guide.* Conservation Guide: DOCAM.

———. 2010d. “DOCAM Documentation Model.” Accessed November 13, 2023. https://www.docam.ca/en/documentation-model.html.

Dominguez Rubio, Fernando. 2020. *Still Life: Ecologies of the Modern Imagination at the Art Museum:* The University of Chicago Press.

Doren, Nina van, and Alexandre Michaan. 2016. “CD-ROM Archiving.”.

Dourish, Paul. n.d. “Paul Dourish's Website: Research.” Accessed November 30, 2023. https://www.dourish.com/research.html.

———. 2017. *The Stuff of Bits:* Massachusetts Institute of Technology. Accessed November 13, 2023. http://library.memoryoftheworld.org/#/book/34d022a6-879b-494e-a1d5-53744c35fe6e.

DPC. 2019. “Executive Guide on Digital Preservation.” Accessed November 13, 2023. https://dpconline.org/our-work/dpeg-home.

E.C.C.O. European Confederation of Conservator-Restorers Organisations. 2003. *E.C.C.O. Professional Guidelines (II): Code of Ethics*. Accessed November 13, 2023. https://www.ecco-eu.org/wp-content/uploads/2021/03/ECCO_professional_guidelines_II.pdf.

École supérieure des arts décoratifs (esads), Strasbourg, Espace multimédia gantner, Bourogne, Vidéo Les Beaux Jours, Strasbourg, Bern University of the Arts (bua), and House of Electronic Arts Basel. 2010. “Digital Art Conservation Project.” Accessed November 01, 2023. https://web.archive.org/web/20230131074807/www.digitalartconservation.org/.

Engel, Deena, and Joanna Phillips. 2018. “Introducing ‘Code Resituation’: Applying the Concept of Minimal Intervention to the Conservation Treatment of Software-Based Art.” In *AIC's 46th Annual Meeting*. EMG review, 1-24: AIC. http://resources.conservation-us.org/emg-review/volume-5-2017-2018/engel-2/. Accessed November 13, 2023.

———. 2019. “Applying Conservation Ethics to the Examination and Treatment of Software- and Computer-Based Art.” *Journal of the American Institute for Conservation* 58 (3).

Engel, Deena, Joanna Phillips, Lauren Hinkson, and Marion Thain. 2018. “Reconstructing Brandon (1998-1999): A Cross-Disciplinary Digital Humanities Study of Shu Lea Cheang’s Early Web Artwork.” *Digital Humanities Quarterly* 12 (2): 1–41. http://www.digitalhumanities.org/dhq/vol/12/2/000379/000379.html#.

Ensom, Tom. 2018. “TECHNICAL NARRATIVES: ANALYSIS, DESCRIPTION and REPRESENTATION in the CONSERVATION of SOFTWARE-BASED ART.” Faculty of Arts and Humanities, King's College London. Accessed October 19, 2023. https://kclpure.kcl.ac.uk/portal/en/studentTheses/technical-narratives.

Ensom, Tom, and Jack McConchie. since 2021. “Preserving Immersive Media Knowledge Base.” Accessed December 01, 2023. https://pimkb.gitbook.io/pimkb/.

Espenschied, Dragan, and Lyndsey J. Moulds. 2019. “An Overview of Preservation Activity for Net Art Anthology.” In *the Art Happens Here: Net Art Anthology*, edited by Michael

Connor, Aria Dean, Michael J. Connor, and Dragan Espenschied, 433–35. New York: RHIZOME.

Espenschied, Dragan, and Klaus Rechert. 2018. "Fencing Apparently Infinite Objects: Defining Productive Object Boundaries for Performative Digital Objects." In iPres 2018 2018, 1–4.

Espenschied, Dragan, Oleg Stobbe, Thomas Liebetraut, and Klaus Rechert. 2016. "Exhibiting Digital Art via Emulation. Boot-to-Emulator with the EMiL Kiosk System." In ipres 2016 2016.

Evens, Tom, and Laurence Hauttekeete. 2011. "Challenges of Digital Preservation for Cultural Heritage Institutions." *Journal of Librarianship and Information Science* 43 (3): 157–65. doi:10.1177/0961000611410585.

Falcao, Patricia, Alistair Ashe, and Brian Jones. 2014. "Virtualisation as a Tool for the Conservation of Software-Based Artworks." In *IPres 2014: Proceedings of the 11th International Conference on Digital Preservation*, edited by ipres 2014.

Falcão, Patricia. 2010. "Developing a Risk Assessment Tool for the Conservation of Software-Based Artworks." Master's Thesis, Conservation, Hochschule der Künste Bern.

Feenberg, Andrew. 1991. *Critical Theory of Technology // Critical Theory of Technology.*). New York, Oxford: Oxford University Press.

———. 2005. "Critical Theory of Technology: An Overview." *Tailoring Biotechnologies* 1 (1): 47–64.

———. 2017. *Technosystem: The Social Life of Reason.* Cambridge, Massachusetts and London, England: Harvard University Press.

Fielding, Roy T. 2000. "Architectural Styles and the Design of Network-Based Software Architectures." Information and Computer Science, University of California, Irvine. Accessed November 13, 2023. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Gassert, Doris. 2013. "Case Study: TV-Bot 1.0 (2004) And TV-Bot 2.0 (2010) By Marc Lee." In *Digital Art Conservation: Preservation of Digital Art. Theory and Practice*, edited by Bernhard Serexhe. 1st ed., 384–403. Wien, Karlsruhe: Ambra V.

Gauër, Nathan. since 2017. "GSOC 2017 | Virgl Windows Driver." Accessed December 11, 2023. https://gist.github.com/Keenuts/199184f9a6d7a68d9a62cf0011147c0b.

Gaure, Teal. 2021. "Hardware and Operating System Configuration of the Permanent Installation of Shan Shui at the Dortmunder U: Description in Nix Package Manager." Accessed December 11, 2023. https://gitlab.com/shanshui/nixos/-/blob/master/flake.nix.

Ghezzi, Carlo, Mehdi Jazayeri, and Dino Mandrioli. 2003. *Fundamentals of Software Engineering.* 2nd. New Delhi: Prentice Hall of India.

Giannachi, Gabriella. 2022a. "The Use of Documentation for Preservation and Exhibition: The Cases of SFMOMA, Tate, Guggenheim, MOMA, and LIMA." In Dekker and Giannachi 2022, 133–44.

———. 2022b. “Unfold: Dan Graham’s Audience/Performer/Mirror Reenacted.” In *on Reenactment: Concepts, Methodologies, Tools*, edited by Cristina Baldacci and Susanne Franco, 3-18. Mimesis journal books. Torino: Accademia University Press.

Git community. n.d. “Git Website.” Accessed November 25, 2023. https://git-scm.com/.

Goodman, Nelson. 1976. *Languages of Art:* Hackett Pub Co Inc.

Google. 2019. “Google Search: Useful Responses Take Many Forms.” Accessed November 13, 2023. https://web.archive.org/web/20190323014825/https://www.google.com/search/howsearchworks/responses/#?modal_active=none.

Grace, Stephen. 2009. “Investigating the Significant Properties of Electronic Content over Time: InSPECT Final Report.” https://web.archive.org/web/20120617000806/http://www.significantproperties.org.uk/inspect-finalreport.pdf.

Grindley, Neil. 2015. *The Digital Curation Sustainability Model (DCSM)*. Accessed November 13, 2023. https://www.4cproject.eu/dcsm/.

Groys, Boris. 1996. “The Restoration of Destruction.” *CAHIER* (4). https://www.fkawdw.nl/en/our_program/publications/cahier_4. Accessed November 13, 2023.

Guttenbrunner, Mark, and Andreas Rauber. 2012. “Evaluating Emulation and Migration: Birds of a Feather?” In *Outreach of Digital Libraries: A Globalized Resource Network : 14th International Conference on Asia-Pacific Digital Libraries, ICADL 2012, Taipei, Taiwan, November 12-15, 2012 : Proceedings / Hsin-Hsi Chen, Gobinda Chowdhury (Eds.)*. Vol. 7634, edited by Hsin-Hsi Chen and G. G. Chowdhury, 158–67. LNCS sublibrary. SL 3, Information systems and application, incl. Internet/Web and HCI 7634. Berlin, New York: Springer.

Head, Martin J., Will Steffen, David Fagerlind, Colin N. Waters, Clement Poirier, Jaia Syvitski, Jan A. Zalasiewicz et al. 2021. “The Great Acceleration Is Real and Provides a Quantitative Basis for the Proposed Anthropocene Series/Epoch.” *Episode* 45 (4). doi:10.18814/epiiugs/2021/021031.

Heinich, Nathalie. 2014. “Practices of Contemporary Art: A Pragmatic Approach to a New Artistic Paradigm.” In *Artistic Practices: Social Interactions and Cultural Dynamics*, edited by Tasos Zembylas, 32–43. Routledge/European Sociological Association studies in European societies 20. London: Routledge.

Hershman Leeson, Lynn. n.d. “Cyberoberta (1995-1998).” Accessed November 29, 2023. https://sites.rhizome.org/anthology/cyberoberta-live.html.

Himmelsbach, Sabine. 2022. “From the Early Web to Web 3.0: On Virtual.Hek.Ch in 2022.” Accessed December 12, 2023. https://cms.hek.ch/files/downloads/220905_virtual.hek.ch_PM_EN.pdf.

Hölling, Hanna. 2016. “Revisions, or on the Aesthetics of Change: Zen for Film.” https://mefsite.wordpress.com/2016/06/07/revisions-or-the-aesthetics-of-change/. Accessed November 13, 2023.

Hölling, Hanna B. 2013. “Versions, Variations, and Variability: Ethical Considerations and Conservation Options for Computer-Based Art.” In *Electronic Media Review (Volume 2)*. Electronic Media Review 2.

———. 2015. “The Archival Turn: Toward New Ways of Conceptualising Changeable Artworks.” In *Data Drift: Archiving Media and Data Art in the 21st Century*, edited by Rasa Šmite, Raitis Šmits, and Lev Manovich. Acoustic space no. 14. Riga, Latvia, Liepāja, Latvia: RIXC, The Centre for New Media Culture; MPLab, Art Research Lab of Liepāja University.

House of Electronic Arts Basel. n.d.a. “HEK Website: About HEK / Mission Statement.” Accessed November 30, 2023. https://hek.ch/en/ueber-das-hek/#leitbild.

———. n.d.b. “HEK Website: Collection.” Accessed December 04, 2023. https://www.hek.ch/en/collection.

———. 2019. “LAN. TraceNoizer - Disinformation on Demand: Artwork Description on HeK Collection Website.” Accessed December 01, 2023. https://hek.ch/en/collection/artworks/tracenoizer---disinformation-on-demand/.

Hughes, Thomas Parke. 1987. “The Evolution of Large Technological Systems.” In *the Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*, edited by Wiebe E. Bijker, Thomas Parke Hughes, Trevor, Pinch, and Deborah G. Douglas: MIT Press. https://bibliothek.wzb.eu/pdf/1986/p86-9.pdf.

Hummelen, Ijsbrand, and Dionne Sillé, eds. 1999. *Modern Art: Who Cares? An Interdisciplinary Research Project and an International Symposium on the Conservation of Modern and Contemporary Art.* Amsterdam: Foundation for the Conservation of Modern Art.

ICOMOS 2nd International Congress. 1964. “Venice Charter.”.

INCCA International Network for the Conservation of Contemporary Art. 1999. “The Decision-Making Model for the CONSERVATION and RESTORATION of MODERN and CONTEMPORARY ART.” In *Modern Art: Who Cares? An Interdisciplinary Research Project and an International Symposium on the Conservation of Modern and Contemporary Art*, edited by Ijsbrand Hummelen and Dionne Sillé, 164–72. Amsterdam: Foundation for the Conservation of Modern Art.

Internet Archive. n.d. “Internet Archive Website.” Accessed December 01, 2023. https://archive.org/.

Ippolito, Jon, and Caitlin Jones, eds. 2003. *The Variable Media Approach: Permanence Through Change:* Guggenheim Museum; Daniel Langlois Foundation.

ipres 2016, ed. 2016. *IPRES 2016 - 13th International Conference on Digital Preservation*. Accessed November 13, 2023. https://zenodo.org/records/1255965.

iPres 2018, ed. 2018. *IPres 2018*. Accessed November 13, 2023. https://osf.io/u5w3q/.

iPres2019, ed. 2019. *IPres2019: Conference Proceedings.* Amsterdam. Accessed November 13, 2023. https://ipres2019.org/program/proceedings/.

ISO/IEC. 2012. *Space Data and Information Transfer Systems — Audit and Certification of Trustworthy Digital Repositories*, no. ISO 16363:2012: ISO. Accessed November 13, 2023. https://www.iso.org/standard/56510.html.

———. 2015. *Information Technology — Vocabulary*, no. ISO/IEC 2382:2015(en): ISO/IEC. Accessed November 13, 2023. https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v2:en.

———. 2022. *Software Engineering — Software Life Cycle Processes — Maintenance*, no. SO/IEC/IEEE 14764:2022.

Janis, Katrin. 2005. *Restaurierungsethik im Kontext von Wissenschaft und Praxis.* Forum Denkmal und Restaurierung 1. München: Meidenbauer. Zugl. Bamberg, Univ., Diss., 2002.

Jennings, Colleen, and Simon Boas. 2018. “Right to RE-.” The Wrong New Digital Art Biennale from Jan 24 to Feb 13, 2018. Accessed December 01, 2023. https://web.archive.org/web/20180811083518/http://right-to-re.xyz/.

Jennings Lewis, Kate, and Tina Weidner. 2013. “GEEKS, BOFFINS, and WHI-KIDS: THE KEY ROLE of the INDEPENDENT EXPERT in TIME□BASED MEDIA CONSERVATION.” *The Electronic Media Review* 2: 119–31. https://resources.culturalheritage.org/emg-review/volume-two-2011-2012/geeks-boffins-and-whiz-kids-the-key-role-of-the-independent-expert-in-time-based-media-conservation/. Accessed November 13, 2023.

JISC. 2008. “Digital Preservation Strategies.” Accessed November 13, 2023. https://wayback.archive-it.org/org-467/20161101115941/http://www.paradigm.ac.uk/workbook/preservation-strategies/selecting-strategy.html.

Jones, Caitlin. 2006. “Conversation with Michael Connor.” Accessed November 13, 2023. https://www.eai.org/resourceguide/exhibition/computer/interview_connor.html.

Keene, Suzanne. 2002. *Managing Conservation in Museums* 2nd ed. Oxford: Routledge. https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=81872&site=ehost-live&scope=site.

KEEP. 2009-2012. “Keeping Emulation Environments Portable (KEEP).” Accessed November 25, 2023. https://www.dnb.de/EN/Professionell/Erhalten/LZA-Historie/_content/lzaHistorieKeep_akk.html.

Kelly, Mat. 2014. “Browser-Based Digital Preservation.” June 7. Accessed November 13, 2023. https://www.slideshare.net/matkelly01/browserbased-digital-preservation?from_action=save.

Kemp, Jonathan. 2022. “Practical Ethics V3.0: Version Control.” In *Kunst Und Material Konzepte, Prozesse, Arbeitsteilungen*, edited by Roger Fayet and Regula Krähenbühl. Outlines 12. [Zürich], Zürich: SIK ISEA; Scheidegger & Spiess.

Khosrow-Pour, Mehdi. 2013. *Dictionary of Information Science and Technology.* Second edition. Hershey PA: Information Science Reference.

Kirschenbaum, Matthew, Richard Ovenden, and Gabriela Redwine. 2010. *Digital Forensics and Born-Digital Content in Cultural Heritage Collections.* Washington D.C.

Kraner, Cem. 1997. “The Impossibility of Complete Testing.” *Software QA magazine* 4 (4): p. 28ff. https://web.archive.org/web/20210813055112/http://www.testingeducation.org/BBST/foundations/Kaner_impossibility.pdf. Accessed November 13, 2023.

Kreymer, Ilya. 2016. Interview by M. McKeehan. 2016. Accessed November 13, 2023. https://ndsr.nycdigital.org/symmetrical-web-archiving-with-webrecorder-a-browser-based-tool-for-digital-social-memory-an-interview-with-ilya-kreymer/.

———. 2020. “Web Object Encapsulation Complexity (Part I).” Accessed November 13, 2023. https://webrecorder.net/2020/11/09/encapsulation-complexity.html.

Kuipers, Marieke, and Wessel d. Jonge. 2017. *Designing from Heritage. Strategies for Conservation and Conversion.* Delft, Netherlands: TU Delft.

Laforet, Anne. 2009. “LA CONSERVATION DU NET ART AU MUSÉE: LES STRATÉGIES Á L'oeuvre.” PhD, Laboratoire Culture et Communication, l’Université d’Avignon.

Latour, Bruno. 1991/1993. *We Have Never Been Modern:* Harvard University Press. translated by Catherine Porter (1993). Accessed November 13, 2023. https://monoskop.org/images/e/e4/Latour_Bruno_We_Have_Never_Been_Modern.pdf.

Laurenson, Pip. 2001. “Developing Strategies for the Conservation of Installations Incorporating Time-Based Media with Reference to Gary Hill's "Between Cinema and a Hard Place".” *Journal of the American Institute for Conservation* 40 (3): 259–66.

———. 2005. “THE MANAGEMENT of DISPLAY EQUIPMENT in TIME-BASED MEDIA INSTALLATIONS.” *Tate Papers*.

———. 2006. “Authenticity, Change and Loss in the Conservation of Time-Based Media Installations.” *Tate Papers*.

———. 2014. “Old Media, New Media? Significant Difference and the Conservation of Software-Based Art.” In *New Collecting: Exhibiting and Audiences After New Media Art / Edited by Beryl Graham, University of Sunderland, UK*, edited by Beryl Graham, 73–96. Surrey (UK) / Burlington (USA): Ashgate Publishing Limited.

Lawson, Louise, and Deborah Potter. 2017. “Contemporary Art, Contemporary Issues? Conservation at Tate.” *Journal of the Institute of Conservation* 40 (2): 121–32. doi:10.1080/19455224.2017.1318079.

Lee, Marc. n.d.a. “Protect Your Databody – Clone-It! Project Description on Marc Lee's Website.” Accessed December 11, 2023. https://marclee.io/en/protect-your-databody-clone-it/.

———. n.d.b. “Tracenoizer – Disinformation on Demand: Project Description on Marc Lee's Website.” Accessed December 01, 2023. https://marclee.io/en/tracenoizer-disinformation-on-demand/.

———. 2004a. “Screencast TV Bot 1.0 (2004): Acquired by the House of Electronic Arts (HEK).” HEK collection. Accessed November 01, 2023. https://www.hek.ch/en/collection/artworks/tv-bot-1-0.

———. 2004b. “Screencast TV Bot 1.0 (First Version 2004): Presented on Marc Lee's Website.” Youtube link: https://www.youtube.com/watch?v=NOHLc0XQ3vI. Accessed November 25, 2023. https://marclee.io/en/tv-bot-world-news-as-soon-as-it-happens/.

———. 2004c. “TV Bot - World News as Soon as It Happens!”. Accessed November 23, 2023. https://marclee.io/en/tv-bot-world-news-as-soon-as-it-happens/.

———. 2010. “Screencast TV Bot 2.0 (First Version 2010): Presented on Marc Lee's Website.” Youtube link https://www.youtube.com/watch?v=LUDxyXanmps. Accessed November 23, 2023. https://marclee.io/en/tv-bot-world-news-as-soon-as-it-happens/.

———. 2021a. “Screenshot from TV Bot 3.0 (2016): Website Http://marclee.Io/tvbot/.” Screenshot taken 2021/01/05 by Claudia Roeck.

———. 2021b. Interview by C. Roeck. January 11, 2021. video conference. Accessed December 21, 2023. https://uvaauas.figshare.com/account/projects/21827/articles/24885933.

Lehman, Meir M., and Juan F. Ramil. 2002. “Software Evolution and Software Evolution Processes.” *Annals of Software Engineering* 14 (1): 275–309. doi:10.1023/A:1020557525901.

Leigh Star, Susan. 1999. “The Ethnography of Infrastructure.” *American Behavioral Scientist* 43 (3): 377–91.

Lialina, Olia. n.d. “My Boyfriend Came Back from the War (1996).” Accessed November 29, 2023. https://sites.rhizome.org/anthology/lialina.html.

———. 2009. “A Vernacular Web.” In *Digital Folklore*, edited by Olia Lialina and Dragan Espenschied, 19-35.

LI-MA. n.d.a. “Catalog Entry and Video Documentation of Dan Graham's Audience/Performer/Mirror (1977): Performed by Dan Graham at De Appel in Amsterdam.” Accessed November 25, 2023. https://www.li-ma.nl/lima/catalogue/art/dan-graham/audience-performer-mirror/62.

———. n.d.b. “LI-MA Website.” Accessed November 29, 2023. https://www.li-ma.nl/.

———. 2016-2017a. “Case Study - Geert Mul, "Horizons" (2008) & "Shan Shui" (2013).” Accessed December 11, 2023. https://li-ma.nl/lima/article/case-study-geert-mul-horizons-2008-shan-shui-2013.

———. 2016-2017b. “Future Proof Media Art: A Digital Art Preservation Project in Collaboration with Geert Mul.” Accessed November 25, 2023. https://www.li-ma.nl/lima/article/future-proof-media-art.

———. 2020. “UNFOLD: Audience/Performer/Mirror.” Accessed November 25, 2023. https://www.li-ma.nl/lima/news/unfold-audienceperformermirror.

Linux Foundation. n.d. “Linux File System Hierarchy Standard.” Accessed November 01, 2023. https://wiki.linuxfoundation.org/lsb/fhs-30.

Lozano-Hemmer, Rafael. 2019. “Best Practices for Conservation of Media Art from an Artist’s Perspective.” In *Digital Art Through the Looking Glass: New Strategies for Archiving, Collecting and Preserving in Digital Humanities*, edited by Oliver Grau, Janina HOTH, and Eveline Wandl-Vogt, 105–16: Edition Donau-Universität.

Lurk, Tabea. 2008. “Virtualisation as a Conservation Measure: Contribution to the Handling of Born Digital Media Art.” In *Archiving 2008: Final Program and Proceedings*, edited by Society for Imaging Science and Technology, 221–25.

Magnin, Emilie. 2016. “Defining an Ethical Framework for Preserving Cory Arcangel’s Super Mario Clouds.” *The Electronic Media Review* 2015-2016: 1–30. https://resources.culturalheritage.org/emg-review/volume-4-2015-2016/magnin/. Accessed November 13, 2023.

McCabe, Thomas J. 1976. “A Complexity Measure.” *IEEE Transactions on software engineering* SE-2 (4): 308–20. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1702388. Accessed November 13, 2023.

McCallum, Andrew. 1994 / 2002. “Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering.” Accessed December 11, 2023. https://www.cs.cmu.edu/~mccallum/bow/.

Meadows, Donella, Dennis Meadows, Jorgen Randers, and William Behrens. 1972. *The Limits to Growth: A Report for the Club of Rome's Project on the Predicament of Mankind.* New York: Universe Books.

Miksa, Tomasz, Rudolf Mayer, and Andreas Rauber. 2015. “Ensuring Sustainability of Web Services Dependent Processes.” *IJCSE* 10 (1/2): 70–81. doi:10.1504/IJCSE.2015.067058.

MoMA, Tate, SFMOMA, and Hellar Studios LLC. n.d. “Guidelines for the Care of Media Artworks.” Accessed November 25, 2023. http://mattersinmediaart.org/.

Monfort, Nick. 2008. “Obfuscated Code.” In *Software Studies / a Lexicon*, edited by Matthew Fuller, 193–99. London: The MIT Press.

Mul, Geert. 2017a. Interview by R. Somers Miles. January 23, 2017. Schiedam (NL). Accessed October 30, 2023. https://vimeo.com/234667835 Video.

———. 2017b. Interview by C. Roeck. February 13, 2017. Schiedam (NL). Accessed October 30, 2023. https://uvaauas.figshare.com/account/articles/8300171.

Mul, Geert, Roelie Zijlstra and Sandra Kisters. 2017. Interview by C. Roeck. August 15, 2017. Rotterdam. Accessed October 30, 2023. https://uvaauas.figshare.com/account/articles/8300309.

Museum Boijmans van Beuningen. n.d. *Jaarverslag 2008 - Museum Boijmans Van Beuningen.* Rotterdam. Accessed November 13, 2023. https://www.boijmans.nl/jaarverslagen-2.

Navrat, Pavol, and Roman Filkorn. 2005. “A Note on the Role of Abstraction and Generality in Software Development.” *Journal of Computer Science* 1 (1): 98–102.

Neddam, Martine. n.d. “MOUCHETTE in ‘Why Not Sneeze?’ (1997).” Accessed November 29, 2023. https://sites.rhizome.org/anthology/mouchette.html.

Nixos. n.d.a. “Nix Package Repository.” Accessed December 11, 2023. https://search.nixos.org/packages.

———. n.d.b. “Nix Reference Manual: Nix Language.” Accessed December 11, 2023. https://nixos.org/manual/nix/stable/language/.

Osterweil, Leon. 2008. “What Is Software?” *Automated Software Engineering*, no. 15: 261–73. doi:10.1007/s10515-008-0031-y.

Owens, Trevor. 2018. *The Theory and Craft of Digital Preservation // the Theory and Craft of Digital Preservation.* Baltimore: Johns Hopkins University Press.

Pagani, Margherita, ed. 2009. *Encyclopedia of Multimedia Technology and Networking // Encyclopedia of Multimedia Technology and Networking.* 2nd ed. Hershey, New York: Information Science Reference. Accessed December 01, 2023. https://doc.lagout.org/Others/ISR.Encyclopedia.Of.Multimedia.Technology.And.Networking.2nd.Edition.Sep.2008.eBook-ELOHiM.pdf.

Pericles. n.d. *Acting on Change: Model--driven Management of Evolving Digital Ecosystems.* London. White Paper. Accessed November 13, 2023. https://web.archive.org/web/20170907103800/http://www.pericles-project.eu/publications/113.

Phillips, Joanna, and Deena Engel, eds. 2023. *Conservation of Tme-Based Media Art:* Routledge.

Phillips, Joanna, Deena Engels, Emma Dickson, and Jonathan Farbowitz. 2017. “Restoring Brandon. Shu Lea Cheang's Early Web Artwork.” Accessed November 13, 2023. https://www.guggenheim.org/blogs/checklist/restoring-brandon-shu-lea-cheangs-early-web-artwork.

Portnoy, Matthew. 2012. *Virtualization Essentials.* First edition, Kindle. Indianapolis: Sybex.

Prelz, Carlo. 2017a. Interview by C. Roeck. March 12, 2017. Bern (Switzerland). Summary of 1st Interview.

———. 2017b. Interview by C. Roeck. May 21, 2017. Bern (Switzerland). Summary of 2nd Interview.

Rancour, Vile. n.d. “FFmpeg CRT Transform: Source Code Repository.” https://github.com/viler-int10h/FFmpeg-CRT-transform. Accessed December 01, 2023.

READ_ME Festival. 2002. “READ_ME Festival 1.2.” Accessed November 01, 2023. https://web.archive.org/web/20030123040946/http://www.macros-center.ru/read_me/inde2.htm.

Rechert, Klaus, and et al. 2019. “Emulation-as-a-Service: Source Code.” Accessed December 01, 2023. https://gitlab.com/emulation-as-a-service.

Rechert, Klaus, Thomas Liebetraut, Dennis Wehrle, and Euan Cochrane. 2016. “Preserving Containers – Requirements and a Todo-List.” In *Digital Libraries: Knowledge,*

Information, and Data in an Open Access Society : 18th International Conference on Asia-Pacific Digital Libraries, ICADL 2016, Tsukuba, Japan, December, 7-9, 2016, Proceedings / Atsuyuki Morishima, Andreas Rauber and Chern Li Liew (Eds.), edited by Atsuyuki Morishima, Andreas Rauber, and Chern L. Liew, 225–30. LNCS Sublibrary: SL3 - Information Systems and Applications, incl. Internet/Web, and HCI 10075. Cham: Springer.

Rechert, Klaus, Dirk von Suchodoletz, and Randolph Welte. 2010. "Emulation Based Services in Digital Preservation." In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, edited by Jane Hunter. New York, NY: ACM.

Rechert, Klaus, Isgandar Valizada, Dirk von Suchodoletz, and Johann Latocha. 2012. "BwFLA – a Functional Approach to Digital Preservation." *PIK - Praxis der Informationsverarbeitung und Kommunikation* 35 (4). doi:10.1515/pik-2012-0044.

Rescience C. n.d. "ReScience C: Journal." Accessed November 25, 2023. https://rescience.github.io/.

Rhizome. n.d. "Rhizome Website." Accessed November 29, 2023. https://rhizome.org/.

———. since 2016. "Net Art Anthology." Accessed November 30, 2023. https://anthology.rhizome.org/.

Riegl, Alois. 1903. "The Modem Cult of Monuments: Its Essence and Its Development: Translation from German to English." 69–83.

———. 1982. "The Modern Cult of Monuments: Its Character and Its Origin: Translated by Kurt W. Forster and Diane Ghirardo in 1982. Source from 1903." *Oppositions. A journal for ideas and criticism in architecture.* (25). https://archive.ph/20140214020632/http://www.modernism101.com/eisenmann_oppositions_set.php. Accessed November 13, 2023.

Rinehart, Richard. 2007. "The Media Art Notation System: Documenting and Preserving Digital/Media Art." *Leonardo* 40 (2): 181–87.

Robillard, Martin P. 2016. "Sustainable Software Design." In *FSE'16: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering : November 13-18, 2016, Seattle, WA, USA*, edited by Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su, 920–23. New York, New York: Association for Computing Machinery.

Roeck, Claudia. 2023. "Languages of Conservation: A Comparison Languages of Conservation: A Comparison: Between Internet-Based Art and Built Heritage." In *Conservation of Contemporary Art: Bridging the Gap Between Theory and Practice*, edited by Renée van de Vall and Vivian van Saaze, 101–25. [S.l.]: SPRINGER INTERNATIONAL PU.

Roeck, Claudia, and Teal Gaure. 2021a. "Wiki for the Documentation of the Permanent Installation of Shan Shui at the Dortmunder U." Accessed November 24, 2023. https://gitlab.com/shanshui/boij/-/wikis/home.

———. 2021b. "Wiki for the Documentation of the Permanent Installation of Shan Shui at the Dortmunder U: Adaptation of Video Resolution and Aspect Ratio." Accessed

December 11, 2023. https://gitlab.com/shanshui/boij/-/wikis/Changes-2021-for-Dortmunder-U#adaptation-of-video-resolution-and-aspect-ratio.

———. 2021c. “Wiki for the Documentation of the Permanent Installation of Shan Shui at the Dortmunder U: New Operating System NixOS.” Accessed December 11, 2023. https://gitlab.com/shanshui/boij/-/wikis/Changes-2021-for-Dortmunder-U#new-operating-system-nixos.

Roeck, Claudia, Klaus Rechert, and Julia Noordegraaf. 2018. “Evaluation of Preservation Strategies for an Interactive, Software-Based Artwork with Complex Behavior Using the Case Study Horizons (2008) By Geert Mul.” In iPres 2018 2018, 1–11.

Roeck, Claudia, Klaus Rechert, Julia Noordegraaf, and Rafael Gieschke. 2019. “PRESERVATION STRATEGIES for an INTERNET-BASED ARTWORK YESTERDAY, TODAY and TOMORROW.” In iPres2019 2019, 179–90.

Rothenberg, Jeff. 1998. “Ensuring the Longevity of Digital Information.” *International Journal of Legal Information* 26 (1-3): 1–22. https://www.clir.org/wp-content/uploads/sites/6/ensuring.pdf. Accessed November 13, 2023.

———. 1999. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation a Report to the Council on Library and Information Resources.* Washington DC: Council on Library and Information Resources.

———. 2000. *An Experiment in Using Emulation to Preserve Digital Publications.* The Hague, Holland: The Koninklijke Bibliotheek; RAND-Europe.

Saaze, Vivian van. 2011. “Acknowledging Differences: A Manifold of Museum Practices.” In Scholte and Wharton 2011, 249–55.

———. 2013. *Installation Art and the Museum: Presentation and Conservation of Changing Artworks:* Amsterdam University Press. Dissertation.

Saaze, Vivian van, Glenn Wharton, and Leah Reisman. 2018. “Adaptive Institutional Change: Managing Digital Works at the Museum of Modern Art.” *Museum and Society* 16 (2). doi:10.29311/mas.v16i2.2774.

Sarff, Michale, and Tim Whidden. 1997. “Simple Net Art Diagram.” Accessed November 13, 2023. http://www.mtaa.net/mtaaRR/off-line_art/snad.html.

Scholte, Tatja, and Glenn Wharton, eds. 2011. *Inside Installations: Theory and Practice in the Care of Complex Artworks / Editors, Tatja Scholte and Glenn Wharton.* Amsterdam: Amsterdam University Press.

Serexhe, Bernhard, ed. 2013. *Digital Art Conservation: Preservation of Digital Art. Theory and Practice.* 1st ed. Wien, Karlsruhe: Ambra V.

Simpson, Justin, Matthew Addis, Jack O’Sullivan, Carl Wilson, Sarah Romkey, and Jon Tilbury. 2019. “Preservation Action Rules Workshop.” In iPres2019 2019.

Smith, James E., and Ravi Nair. 2005. *Virtual Machines: Versatile Platforms for Systems and Processes / James E. Smith, Ravi Nair.* Amsterdam, London: Morgan Kaufmann Publishers.

Software Heritage Team. n.d. “Software Heritage Website.” Accessed November 25, 2023. https://www.softwareheritage.org/.

Software Sustainability Institute. n.d. “Online Sustainability Evaluation.” Accessed December 01, 2023. https://www.software.ac.uk/resources/online-sustainability-evaluation.

———. 2011. “Approaches to Software Sustainability.” Accessed November 13, 2023. https://web.archive.org/web/20220928123331/https://software.ac.uk/resources/approaches-software-sustainability.

Sompel, Herbert van de, Martin Klein, and Shawn Jones. 2021. “Robustifying Links to Combat Reference Rot.” (50). https://journal.code4lib.org/articles/15509.

Stakem, Patrick. 2017. “Graphics Processing Units, an Overview.” *Computer Architecture Series* 16.

Strate, Lance. 2010. “Korzybski, Luhmann, and McLuhan.” *Proceedings of the Media Ecology Association* 11: 31–42.

Strodl, Stephan, Christoph Becker, Robert Neumayer, and Andreas Rauber. 2007. “How to Choose a Digital Preservation Strategy: Evaluating a Preservation Planning Procedure.” In *Proceedings of the 7th ACMIEEE-CS Joint Conference on Digital Libraries*, edited by Edie Rasmussen. New York, NY: ACM.

Suber, Peter. 1988. “What Is Software?” *Journal of Speculative Philosophy* 2 (2): 89–119. https://legacy.earlham.edu/~peters/writing/software.htm. Accessed November 14, 2023.

Suchodoletz, Dirk von. 2010. “Future Emulation and Automation Research Agenda.” *Dagstuhl Seminar Proceedings. Automation in Digital Preservation*.

The National Digital Stewardship Alliance. 2019. “NDSA Levels of Digital Preservation.” Accessed November 14, 2023. https://ndsa.org/publications/levels-of-digital-preservation/.

The Software Preservation Network. n.d.a. “Emulation-as-a-Service Infrastructure: Website.” Accessed December 01, 2023. https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/.

———. n.d.b. “Website of the Software Preservation Network.” Accessed November 25, 2023. https://www.softwarepreservationnetwork.org/.

———. 2019, ongoing. “EaaSI User Handbook.” Accessed November 25, 2023. https://eaasi.gitlab.io/eaasi_user_handbook/.

Thommen, Fabian, Marc Lee and Annina Ruest. 2017. Interview by C. Roeck. June 22, 2017. Zurich (Switzerland). Accessed October 30, 2023. https://uvaauas.figshare.com/articles/media/_/8300408.

Udoh, Emmanuel. 2009. “Open Source Database Technologies.” In Pagani 2009, 1106–11.

United Nations. 2014. “Report of the Open Working Group of the General Assembly on Sustainable Development Goals.” Unpublished manuscript, last modified November 14, 2023. https://digitallibrary.un.org/record/784147?ln=en.

V2 Rotterdam. 2000. “Creative Kernels: Workshop for the Presentation and Discussion of Open Source Projects.” Accessed December 12, 2023. https://v2.nl/events/creative-kernels.

Vall, Renée van de. 2017. “Documenting Dilemmas. On the Relevance of Ethically Ambiguous Cases.” *Sin Objeto.* doi:10.18239/sinobj_2017.00.0.

Variable Media Network. n.d. “The Variable Media Network Website.” Accessed November 29, 2023. https://www.variablemedia.net/e/index.html.

———. 2004. “Seeing Double: Emulation in Theory and Practice.” Accessed November 21, 2023. https://www.variablemedia.net/e/seeingdouble/.

VDE-community. n.d. “Virtual Distributed Internet (VDE) Library.” Accessed December 01, 2023. https://github.com/virtualsquare/vde-2.

VirGL community. since 2018. “VirGL: The Virtual GPU Project (Open Source GPU Driver for Virtual Machines).” Accessed December 11, 2023. https://virgil3d.github.io/.

W3 Schools. n.d. “HTML IFrames.” Accessed December 01, 2023. https://www.w3schools.com/html/html_iframe.asp.

Wagner, Christian. 2014. *Model-Driven Software Migration: A Methodology: Reengineering, Recovery and Modernization of Legacy Systems.* Aufl. 2014. Springer eBook Collection / Computer Science. Wiesbaden: Springer Fachmedien Wiesbaden.

Webrecorder community. n.d. “Https://github.Com/Webrecorder/pywb: Python Web Archiving Tools (Pywb).” Accessed November 01, 2023. https://github.com/Webrecorder/pywb.

Wijers, Gaby. 2011. “To Emulate or Not: CONSERVATION CASE STUDIES from the NETHERLANDS.” In Scholte and Wharton 2011.

Wilson, Andrew. 2005. “A Performance Model and Process for Preserving Digital Records for Long-Term Access.” Archiving 2005 Final Program and Proceedings. *Society for Imaging Science and Technology*, 20–25. Archiving 2005 Final Program and Proceedings.

———. 2007. “Significant Properties Report: InSPECT Work Package 2.2.” Version 2.

Zenodo. n.d. “Zenodo Website.” Accessed November 25, 2023. www.zenodo.org.

Zhang, Ga. 2020/2021. “We=Link: Sideways: Online Exhibition.” Accessed December 12, 2023. https://web.archive.org/web/20220318172159/http://we-link.chronusartcenter.org/.

Appendix

A.1 Matrix Digital Materiality

The table is to be filled in with text respectively content, not just with values, as it is often not possible, to assign values. The table is just a tool to become aware of the different aspects of digital materiality. The table can naturally be extended with other values listed by (Appelbaum 2007, 89). Examples of a filled in matrix see following pages.

Artwork	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Newness value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**[74]
Artwork components / layers / digital material[75]						
(Wider) Artwork environment						
Artwork as a whole						

[74] The art value does not necessarily depend on the fact, whether a component is made by the artist. I also assume, that the artwork as a whole always has an art value.
[75] Digital material such as programming languages, software, software libraries, operating system, digital file formats if important for the artwork

Digital materiality of *Horizons*

***Horizons* (2008)**[76]	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Newness value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Software and software environment						
Scripts in Ruby and C Version (2013)	Carlo Prelz programmed for artists in the context of V2 and other art labs. His practice was routed in the open-source and artist community in the Netherlands (V2 Rotterdam 2000). For the driver of the tracking sensor, he adapted a script written by Marnix de Nijs. C is often used to program device drivers, operating systems or other programming languages, Ruby is more abstracted (higher-level) than C and typically used to program applications.	High. The C script sick.c is crucial for the interpretation of the sensor input and the Ruby scripts are generating the video and sound output.	High. The scripts create a video and sound based on an image database and on the movement of the visitors.	Intermediate. The work was updated and stabilised since 2008. The work feels ageless, when	Intermediate. The work was updated and stabilised since 2008.	High. Scripts custom-made by Carlo Prelz in collaboration with Geert Mul. They represent the core of the work.

[76] 2008 is the year of creation of the artwork. However, version 2008 of Horizons does not exist any longer. This table refers to version 2013 described in section 4.6.

***Horizons* (2008)**[76]	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Newness value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Operating system Debian 7 (2013)	Linux operating system. Common in open-source community.	Low. Indirect contribution.	High.	Low.	Low.	Low.
Hardware (Computer, tracking sensor, video projectors. speakers)	They are all generic pieces of equipment that are massproduced in factories. They are all typical for 2013-2016 (regarding their tech specs and aesthetics). Such video projectors and speakers are often used in exhibition spaces. The computer is a desktop computer in a large case that can be used in a home or business context. The computer hardware is not visible. The tracking sensor, the video projectors and the speakers are visible in the background, but their look is not important. They are all generic pieces of equipment.	Look of the equipment: Low. The computer hardware is not visible. The look of the tracking sensor, the video projectors and the speakers is not important. Impact on sound and video aesthetics: intermediate. Hardware and software are both contributing to the properties of video and sound (performance / speed, aspect ratio of video, video and sound resolution etc).	High.	Low.	Low.	Low.

Horizons (2008)[76]	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Newness value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Environ-ment						
Production facilities of equipment	Often, planned obsolescence and/or the minimisation of electronic components makes repair of equiqment difficult. Thus, replacement of the equipment is often the best solution.	N.a.	Intermediate. Indirect effect. When the equipment factories no longer produce the same models, conservation strategies have to make sure that the artwork runs on new equipment.	N.a.	N.a.	N.a.
Exhibition space	N.a. Depends on the space.	Low. Has to comply to minimal standards (display specifications).	Low. Has to comply to minimal standards (display specifications).	N.a.	N.a.	N.a.

***Horizons* (2008)**[76]	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Newness value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Artwork as a whole						
Artwork as a whole	The code of the artwork and the way of displaying a museum collection of paintings in an automated and interactive way can be seen as a typical idea for the time when the artwork was created.	High. The smooth rendering of the video and the quality of the sound are important to the artist. The reaction of the video and sound output to the visitor movement is important.	High. The reaction of the video and sound output to the visitor movement is important. The fact, that a software controls the output is important. It must not be replaced by a video.	Low. The piece is perceived as quite ageless, as the equipment cannot be seen and the interaction with the tracking sensor is intuitive.	Low to intermediate. The piece was updated in 2013.	High. The code is essential for the artwork and was custom-made.

Digital materiality of *TraceNoizer*

Trace-Noizer	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Web server						
Scripts in PERL 5.4, PHP 4, HTML 4, MYSQL 4	Example of an early dynamic website based on framesets and based on a variation of LAMP (Udoh 2009). PERL, founded in 1987, is used for text processing, which is typical. PERL is still developed in 2023. PHP, founded in 1995, is one of the first web programming languages and is still frequently used in 2023, although it has passed its peak. HTML as the basic web page language went through major developments since 2004. The page does not contain JavaScript which is the main web programming language in 2023.	Direct contribution. The pages are a puzzle of frame sets. Hence, switching from one page of *Tracenoizer* to another does not show a different URLpath. The design of the page is clearly noughties. No adaptation to mobile devices.	Direct contribution. Users can create clones with TraceNoizer and login to manage their clones.	High. Old websites evoke nostalgic feelings. / Examples of old websites are rare, as their software and internet environments disappear.	High. The languages as such are not rare, but their versions are. The fact that *TraceNoizer* represents an example of the web programming culture (including backend) of the noughties is special. The scripts are crucial for the website.	High. Custom-made by artists. Core of the work.

Trace-Noizer	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Unspecific operating system, e.g. Debian 1.3, preserved on the *Trace-Noizer* CD-ROM	Linux operating systems such as Debian were typically used for web servers as they were light, free and open-source. Use of open-source libraries and ethics.	Low. Indirect contribution.	High. Direct contribution.	Low.	Low.	Low.
Unspecific Hardware (Web server computer from 2004 does no longer exist).	Standard web server computer from 2004. Artists hosted the website themselves.	Look of the equipment: low. The web server is not visible. Influence on look of the website: small Indirect contribution	High. Direct contribution. Computer is replaceable.	Low.	Low. This kind of computers can probably be found in computer museums and can still be acquired second hand.	Low.

Trace-Noizer	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Client						
Unspecific web browser from 2004, e.g., Mozilla Firefox 1.3	Web browsers were and are the gate to the internet. In 2004 there was a big competition between web browsers as they were not as standardised as today. However, for *TraceNoizer* no specific plugins were necessary.	Low. Although old web browsers have a different aesthetic than new ones, the web browser does not change the aesthetics of *TraceNoizer* much.	High. A web browser is necessary to render *TraceNoizer*. A modern web browser can render *TraceNoizer* as well, although there are small differences to a web browser from 2004.	Intermediate. Old web browsers evoke nostalgic feeling.	High. Web pages based on old web technology should be viewed in old web browsers.	Low.
Hardware unspecific	See web server.	See web server.	See web server.	See web server.	See web server.	Low.

Trace-Noizer	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Environ-ment						
World Wide Web in 2004. Only exists as archived pages such as in Internet Archive	In 2004, Web 1.0 prevailed. Personal web pages were common, as social media were only beginning. There were fewer dynamic websites. Websites were linked to each other to be discoverable. Interaction between users happened through email, forums, website guest books etc.	Intermediate. The aesthetics of today's WWW does not match the aesthetics of *TraceNoizer* in contrast to the WWW from 2004.	High. The cloning process in *TraceNoizer* works better with old websites than parsing and reassembling today's social media posts.	Intermediate. The Internet is constantly evolving which is why old internet environments are rare.	Intermediate For *TraceNoizer*, the historical internet environment is important for its functionality. However, if it worked with the new internet environment, it would be acceptable.	Low.
Internet as an infrastruc-ture as it was in 2004	Internet was not as ubiquitous back in 2004 as it is today. Percentage of housholds that had internet access in Switzerland (Bundesamt für Statistik Schweiz n.d.): in 2004: 61% in 2022: 99%. Broadband internet access of housholds in Switzerland: in 2006: 53% in 2021: 99%	Low. The generation of clones was slower due to lower internet bandwidth in 2004.	Important, but generic.	N.a.	N.a.	N.a.

Trace-Noizer	**Socio-cultural value**	**Aesthetic value / Behaviour**	**Use value / Functionality**	**Age value / Rarity**	**Historical value**	**Art value / custom-made / made by artist**
Artwork as a whole						
Artwork as a whole	Representative of a web-based artwork of the noughties. Reflecting on online identity and on users leaving traces in the www. Won several media art awards.	Dynamic website. Modest aesthetics of the noughties. Not adaptable to mobile devices.	User can create clones to blur the traces of unwanted content on websites.	Intermediate. Not many artworks from that period still exist. Some are converted to more current web technology; others are partly harvested by the Internet Archive.	Intermediate. Websites from the noughties whose historic backend has been preserved are rare.	High.

Glossary

Definitions and terms used in this dissertation.

ABI Application Binary Interface	The ABI is the "Binary-level interface between application programs and the operating system". "Compiled binary applications can be ported between systems with the same ABI". The Java Virtual Machine is an example of an ABI.	(Butterfield, Kerr, and Ngondi 2016, 1–2)
Abstraction	Abstraction serves to reduce complexity and is a design principle of **software** engineering. "At the hart of software architecture is the principle of abstraction: hiding some of the details of a system through **encapsulation** in order to better identify and sustain its properties. A complex system will contain many levels of abstraction, each with its own architecture."	(Fielding 2000: 5)
Ambiguity	Different meanings and interpretations of a **software** artifact due to its different levels (source code, execution).	Based on (Dekker 2014, 34)
Allographic / autographic	According to the philosopher Nelson Goodman an artwork is "autographic if and only if the distinction between original and forgery of it is significant; (…) Thus painting is autographic, music nonautographic, or allographic." Often allographic works such as theatre plays or concerts are two-stage works. First the composer or dramaturg writes a score, and in a second step the musicians or actors interpret the score.	(Goodman 1976, 113) Based on (Goodman 1976, 114)
Art conservation	Conservation practices in art museums, treating each art object individually, each art object having an idiosyncratic set of **work-defining properties**. See also **conservation** and **work-defining properties**.	
Automation	Characteristic of **software**. **Software** can automate processes and it can be subjected to automated processes such as compilation or updating or cloning.	

Bridge / Bridging	Sub-strategy of the **encapsulation** strategy A bridge is a local patch that abstracts part of an environment of a **software-based artwork** such as a peripheral and isolates the **software** artifact from the changing environment (peripherals). It interfaces between the old and the new peripheral so that the software artifact does not need to be changed if a new peripheral is connected.	
Browser emulation / remote browser	Used when a website requires an obsolete version of a web browser. Instead of presenting the website directly in a modern web browser, an **emulation** of an obsolete computer system with an obsolete web browser is used to access the website. The browser emulation runs remotely. When the user accesses the website in question, the remote computer with the browser emulation is booted automatically. The user then sees the website through an obsolete web browser within his/her own modern web browser window.	
Building block	A **software-based artwork** can consist of several building blocks. The building blocks are defined such that an individual conservation strategy can be applied for each. For instance, a **web-based artwork** usually consists of at least two building blocks: the web server and the web client.	See section 3.7
Classic virtualisation, classic virtual machine	Sub-strategy of the **encapsulation** strategy. In contrast to **emulation**, the guest system has direct access to the host computer's **CPU** and the system commands do not have to be translated from guest to host. This is only possible if the computer architecture of guest and host is compatible. Due to the direct access to the **CPU**, classic virtualisation performs much better than **emulation**.	For instance (Guttenbrunner and Rauber 2012: 162)

Client-side web page	A web page for which all the computations are executed by the web browser on the client computer. This is in contrast to a server-side web page that requires the web server to compute certain client queries. See server-side web page. A client side-web page can be static (no computations, the html files are served as is) or dynamic, responding to user input through javascript scripts that are embedded in html files.	
Code resituation	Sub-strategy of **migration**, based on minimal intervention. "Code resituation, instead, aims to preserve the original artist's code, or significant parts of it, while adding restoration code to reanimate defunct code to full functionality".	(Engel and Phillips 2018: 2)
Complexity of program evolution	The complexity of program evolution is the probability that a) a change hits a given software module and b) that another software module becomes impacted by the change.	Based on (Belady and M.M. Lehman 1985, 336)
Computer	"A device or system that is capable of carrying out a sequence of operations in a distinctly and explicitly defined manner. (…) The definition of the sequence is called the program."	(Butterfield, Kerr, and Ngondi 2016, 105)
Continuum approach to artwork biography	"There is therefore a need for a framework capable of linking together technical documentation and the broader social and historical context of the artwork, in order to be able to effectively capture narratives of **software-based artwork** evolution."	(Ensom 2018: 204)
Concept of continuity	The artwork should not undergo big changes between artwork manifestations, so that "the agreement of sameness" is not threatened.	(Saaze 2013, 105).

Conservation, preservation	Conservation and preservation both include the whole range of actions between preventive conservation that changes the environment of the object but not the object directly, and conservation interventions that change the object significantly, including restoration. Conservation is a term rather used for art conservation in museums. Preservation is a term rather used for the conservation of physical and digital documents in archives and libraries. In this dissertation, conservation and preservation are used interchangeably. Conservation / preservation "seeks to secure the life that already is"	 Butler in (Castriota 2019a: 44).
Conservation cycle	Several conservation cycles within a life cycle of a **software-based artwork**.	
Conservation measure / intervention	Conservation measures or interventions implement conservation strategies.	
Conservation strategy / preservation strategy	Conservation strategies in this dissertation comprise: • **Storage** • **Documentation** • **Migration** • **Encapsulation** • **Reconstruction** Other conservation strategies: • **Continuation / Cultivation** • **Reinterpretation**	See section 6.3
Conservator	A conservator cares for and preserves the artwork. Member of the **network of care** (see "**network of care**").	

Contemporary art	Contemporary art is a game with boundaries. "The most important transgression of the usual criteria defining art is that the artwork is no longer exclusively the actual object proposed by the artist, but rather, the whole set of operations, actions, interpretations, etc., brought about by this proposition." "Transgressing the boundaries of art also means using new kinds of materials or modes of presentation. Installations, performances, land art, body art, video, large-scale color photographs, multimedia, net art, have become the basic vocabulary of contemporary artists."	(Heinich 2014, 34) (Heinich 2014: 35) (Heinich 2014, 36)
Container	Sub-strategy of the **encapsulation** strategy. A container is an operating system **virtualisation**. Thus, a container depends on the operating system of the host (Guest OS must be compatible with kernel of host OS).	
Continuation / Cultivation	**Conservation Strategy**. Continuation of a processual artwork by adding content. "The process is assumed to be the core of the work and the main aim of conservation is support of the work's continuation through transmission of the required information, skills and procedures to the designated participants or stakeholders." See also term "**Safeguarding**". According to the Software Sustainbalitiy Institute, "cultivation is the process of opening development of the **software**".	(Vall 2017, 85) (Software Sustainability Institute 2011)
Core properties	s. "**Structure with a permeable core**"	
CPU, Central Processing Unit	Term used in computer science: "Functional unit [of a computer] that consists of one or more processors and their internal storages. A processor interprets and executes instructions. CPUs are computer architecture specific. The **abstraction** of a specific CPU is called **ISA** (see **ISA**, instruction set architecture).	(ISO/IEC 2015)
Curator	A curator curates an exhibition and selects artworks for the acquisition. Member of the **network of care** (see "**network of care**").	

Cyclomatic complexity	Cyclomatic complexity indicates the complexity of a program by calculating the number of paths through a program and comparing it to the number of nodes. The goal of this metric is to be able to assess whether a **software** module is difficult to test or maintain.	(McCabe 1976, 308)
Damage	A damage is perceived if the artwork does not match the **significant properties**. This can include loss of functionality, an impaired look and feel, etc.	
Dependency	A dependency is a technical dependency of a **software**, such as a library, specific hardware or video card, or a technical standard such as network layers and file formats. If the dependency is broken, the artwork does not function properly. Besides a technical dependency, the artwork can also be dependent on an online performance or other interaction of the artist (or other contributors).	
Documentation strategy	Conservation strategy. Presenting the captured output of a **software-based artwork** or archival material instead of the output generated in real-time. Capturing input and running the artwork based on the captured input is also an option.	Based on (Dekker and Giannachi 2022, 4)
Digital art	Digital art includes not only **software-based art**, but also born digital art based on images, video and audio.	

Digital materiality	"The materialities of information are those properties of representations and formats that constrain, enable, limit, and shape the ways in which those representations can be created, transmitted, stored, manipulated, and put to use (…)." Reflecting the materiality of the digital is a "way of assessing the relationship between the social and the technical". "The materialities of digital systems lie not least, then, in this gap between what is denoted and what is expressed, or between the specification and the execution. A computer program may be a precise series of instructions, and yet the experience that results from the program's execution is radically underspecified by that program.“	(Dourish 2017, 12/190) (Dourish 2017, 38/190) (Dourish 2017, 24/190)
Digital preservation	Digital preservation practices mainly developed in the archival sector handling large numbers of objects with similar properties. See also "**preservation**"	
Emulation	Sub-strategy of the **encapsulation** strategy. The encapsulation layer isolates the software **environment** from the hardware environment and serves as a bridge to the hardware. An emulator is **software** that is able to interpret the machine language commands of the guest computer architecture (**ISA**) and translates the commands in real-time to the host computer architecture. In practice emulators represent a complete computer system, typically including a sound card, graphics or network adapters. Emulation is also called full-system-emulation. Smith and Nair call this a whole system virtual machine. Guttenbrunner and Rauber call this a full hardware emulation.	(Smith and Nair 2005: 20) (Guttenbrunner and Rauber 2012: 161)
Emulator	Software that emulates hardware (see **Emulation**)	

Emulation-as-a-Service (EaaS)	It is a service that provides **emulators** to the client. It supports the modular assembling and reuse of software **environment**s. The service can either be distributed (each client is a node) or it can be centralised (cloud service). An example for a distributed **Emulation**-as-a-Service the **EaaSI** project by US-American universities. The advantage of a distributed service is that software **environments** can be shared amongst the participants (nodes). In both cases (distributed or central), **emulations** can be made accessible online and embedded in other websites.	Emulation as a central service: (Rechert, Suchodoletz, and Welte 2010)
EaaSI	EaaSI is short for Emulation-as-a-Service Infrastructure and refers to the **EaaS** infrastructure project at US-American Universities initiated around 2019. OpenSLX, a start up from the Universty of Freiburg (Germany) built the distributed infrastructure for this service. The purpose of this emulation infrastructure is sharing emulation and software environments for obsolete software artifacts.	Website: (The Software Preservation Network n.d.a)
Encapsulation	**Conservation strategy**. The artwork is protected through a **layer of abstraction** from unintended changes with the goal to preserve the artwork. The layer of **abstraction** can abstract a computer hardware (**emulation and other virtual machines**), a peripheral (**bridge**), or a network (**virtual network**).	
Environment	The closer environment of an artwork such as the soft- and hardware stack can be considered as part of the artwork. The environment consists of the wider environment that provides context to the artwork such as linked websites of a **web-based artwork** but without which the artwork still functions.	
Functionality	Characteristic of **software**. **Software** specifies instructions to implement a functionality.	Based on (Ensom 2018: 60).
Functional layer	See **layer of functionality**	
Generalisation	means to group problems according to similar characteristics and find a common solution for them. Generalisation increases the chances to be able to reuse **software** components.	Based on (Navrat and Filkorn 2005, 100)

GPU Graphical Processing Unit	Also called video card or graphics card. “A GPU is a specialized computer architecture to manipulate image data at high rates. The GPU devices are highly parallel, and specifically designed to handle image data, and operations on that data. They do this much fastest than a programmed general-purpose **CPU**.”	(Stakem 2017, 49%)
Instantiation of an artwork	Instantiation is the same as manifestation. According to Castriota, manifestation refers to a discrete occurrence or instance of a work in time and space; According to DOCAM, a manifestation is a physical embodiment of an intellectual or artistic realisation of the work;	Based on (Castriota 2019b, 74) (DOCAM 2010d)
Institution	In this dissertation, I define institutions as organisations holding and caring for software-based art. This includes art museums, but also other institutions that do not necessarily exhibit their collections regularly or might not describe themselves as museums.	
ISA Instruction Set Architec-ture	The ISA describes the computer architecture. It “marks the division between hardware and **software**”. Smith and Nair call the part of the ISA that is visible to the application program “user ISA” and the part that is visible to the operating system “system ISA”. The system ISA is responsible for managing hardware resources.	Based on (Smith and Nair 2005, 7,8)
Internet Archive	The Internet Archive is a North American institutition harvesting and archiving websites and making them publicly accessible.	Website: (Internet Archive n.d.)
Internet art, internet-based art	Internet art or internet-based art uses the Internet as an artistic medium. Internet art is an overarching term and includes **web-based art** and **net art** as well as networked art depending on other internet protocols than http. See “**net art**” and “**web-based art**”.	

Layer, layer of functionality	**Software-based art** is always built in layers. It consists of a **software** and hardware stack. Networks such as the internet are also based on layers. Usually, a layer abstracts underlying technology and is hierarchical. Layers can be points of intervention for **emulation** and **virtualisation**.	
Liminal materiality	**Software** "simultaneously occupies multiple material states, without definitively belonging to any of them."	(Ensom 2018, 59)
Longevity	Medium life of the versions of a **software-based artwork**	
Lifecycle, life expectancy	The lifecycle of a **software-based artwork** consists of its creation, optimisation and stabilisation, its perpetuation through conservation cycles, its aging, becoming dysfunctional and becoming forgotten or de-accessioned.	
Maintenance	Maintenance consists of a series of actions to keep a **software** operational. Its purpose is to adapt the **software** to a changing **environment**. Updating the artwork software directly is considered as **migration**. Updating an **emulator**, the artwork **environment** or reconstructed parts of the artwork are considered as maintenance, as well as monitoring the necessity to update or the behaviour of the artwork.	Derived from (ISO/IEC 2022)
Media art / time-based media art	Time-based media art includes art based on electronic media such as video-based art, film-based art, **software-based art** etc.	

Migration	**Conservation strategy**. Migration of digital objects encompasses file (or format) migration and **software** migration. The digital object itself is changed and adapted to the new **environment**. **Software** migration means the transfer of a software artifact to an updated or different software **environment** in order to sustain its functionality. It may include manual adaptations of the software to the new environment or software updates distributed by software developpers, and it might require recompilation of the source code or installation of additional software components such as libraries and drivers.	
Multiplicity	Characteristic of **software**. Due to the fact that it can be duplicated so easily, different versions can exist at the same time.	Based on (Ensom 2018: 60).
Net art	In 2001, Tilman Baumgärtel defined Net art as art that "deals with the specific conditions the internet offers. It plays with the protocols of the Internet." Nowadays, the term net art is often used for early **web-based art** (1990ies to early 2000s). See also "**internet art**" or "**internet-based art**" and "**web-based art**"	Tilman Baumgärtel in (Dekker 2014: 27).
Network of care	The network of care, a term coined by Annet Dekker, "is based on a transdisciplinary attitude and a combination of professionals and non-experts who manage or work on a shared project". The network of care is relevant in most museum processes around the artwork: acquisition, preservation, display.	(Dekker 2014, 81).
Opacity	Characteristic of **software**. Processes happening in the background are not visible to the user. Massive changes in the **software** are not necessarily visible on the user interface.	Based on (Ensom 2018: 60)
Paravirtuali-sation	Sub-strategy of the **encapsulation** strategy. Paravirtualisation offers direct hardware access (CPU and GPU) which improves its performance. Through abstraction of the video card, it can access the GPU.	
Preservation	See "Conservation"	

Reconstruction	Conservation strategy. With a reconstruction one tries to achieve the same look, behaviour and/or function of an art object, without necessarily using the same material or the same structure. Reprogramming a **software functionality** with a different programming language is a reconstruction.	Based on (DOCAM 2010b)
Refactoring	Sub-strategy of **migration.** The code is restructured to improve its readability and reduce maintenance while leaving its behaviour unchanged.	Based on (Bowers et al. 2002, 108)
Reinterpre-tation	**Conservation strategy**. Reinterpretation is an artistic strategy that can also be used for conservation. Reinterpretation has to convey the idea of the artwork free from the material and aesthetic form of the artwork.	Based on (Ippolito and C. Jones 2003, 128)
Ruin	Heavily damaged artwork. Functionality and / or look and feel are impaired.	
Remote browser	See **browser emulation**.	
Reprogram-ming	**Conservation strategy**. Reprogramming can be subsummed under **reconstruction**. To reconstruct a **software functionality** with a different software or pogramming language than before.	
Runtime (short for runtime system or runtime environment)	**Software** and hardware **environment** that enables the execution of a source. The runtime usually consists of several hierarchical layers (a stack). It includes the operating system and libraries, but it can also include a software runtime such as a Java Virtual machine, as well as video and sound cards and their drivers.	
Safeguarding	Safeguarding creates the conditions to allow an artwork to develop further in different directions. See also **Continuation / Cultivation**.	Butler in (Castriota 2019a: 44).

Server-side web page	A web page that requires the web server to compute certain client queries. Such web pages are dynamically composed based on user input. This is in contrast to client-side web pages for which all the computations are executed by the web browser on the client computer. See client-side web page.	
Significant properties	Significant properties are a subset of the **work-defining properties**. They are the properties that are going to be preserved for a specific designated community.	Based on (Grace 2009, 6)
Software	Software runs on a real or virtualised computer hardware or other microcontrollers. It consists of scripts based on algorithms that instruct the hardware what to do. It also comprises digital data. Software exists on different levels of **abstraction** (for instance operating system software, application software, compilers etc.).	
Software-based art	"Artworks where **software** is the primary mechanism in the realisation of the work and the primary material which the artist has chosen as a means of expression". Includes net art.	(Ensom 2018: 18)
Software complexity	The complexity of **software** as defined for this dissertation consists of both a **structural complexity** that alludes to the network of internal and external dependencies as well as the **source code complexity** (complexity within the source code).	
Software performance model	The software performance model assumes that a source needs a technical **environment** to generate a process and that together they create a performance. The process is about what the artwork does (its function). The performance is rather about how the artwork executes the process and about what the visitor/user can experience.	National Archives of Australia (NAA)
Software migration	See **migration**	

Source code complexity	The source code complexity is defined here as the complexity within the source code. Several metrics describe it, as for instance the **cyclomatic complexity** and the **complexity of program evolution.**	
Storage Strategy	**Conservation strategy**. Storage is a a basic strategy, meaning all other conservation strategies are executed in addition to the storage strategy. Storage keeps the object in an unchanged and deactivated state. It aims to prevent detoriation through aging by providing good storage conditions. The components of the artwork are (spatially or physically) not connected to each other and might be stored at different locations. In the field of digital preservation, the storage strategy is also called bit preservation, as the digital objects are preserved on the bit level. Bit preservation makes sure that the content and format of the files do not change, but it does not make sure that the content stays usable and readable. Certain standards apply for bit preservation (for instance (Owens 2018, 71)). In contrast to physical objects, additonal conservation strategies are carried out on duplicates of the stored digital objects, creating new versions.	Based on (Ippolito and C. Jones 2003, 129) Based on (DPC 2019)
Structural complexity	Structural complexity is a characteristic of software. **Software** "presents a complex structure that includes linkages with its **environment**, including external systems and resources."	(Ensom 2018: 60)
Structure with a permeable core	A term used by Castriota to consider the variability of a contemporary artwork. Each manifestation of an artwork creates a set of **significant properties**. The properties of these manifestations are condensing into a critical mass of properties, the core of the artwork. The core is not fixed but can move.	(Castriota 2019b: 171)

Sustainable conservation strategy	**Conservation strategies** are sustainable a) if they ensure access and readability of the artwork and its documentation. b) if they lower or at least do not increase the preservation effort for the next preservation cycle, for instance by reducing the structural complexity of the artwork or by stabilising source code complexity (not increasing it). c) if they are scalable. d) if they allow future generations to preserve the artwork based on new technologies and knowledge. e) if the **network of care** stabilises the **significant properties** of the artwork to a certain extent. f) if the **network of care** documents preservation measures and the decisions in a transparent way. g) if the **institution** is able to provide a conservation budget and policies that enable the perpetuation of the artwork in the long-term.	See section 3.1 (theoretical framework)
Technosystem	"The technosystem is a field of technical practices aimed at control of the environment, whether natural, economic, or administrative. To that end the environment is interpreted and structured as an ensemble of sociotechnically rational functions".	(Feenberg 2017, 159)
User, visitor, viewer, audience, public	People who experience and interact with the artwork. Different terms are used depending on the context.	
Variability	Characteristic of **contemporary art** and **media art.** The space, the exhibition context and other artwork contexts have an influence on the presentation of the artwork.	Based on (Saaze 2011, 249)

Version of an artwork	"Rather than referring to their particular manifested token instances in time and space (…), a version or iteration refers to the various subtypes, which in turn may or may not be tokened in time and space through concrete manifestations or instances."	(Castriota 2019b, 75)
Video card	See "**GPU**"	
Virtualisation	Conservation strategy **encapsulation**. General term including **emulation**, classic virtualisation (classic **virtual machines**) and **containers**.	
Virtual Machine (VM)	In this dissertation overarching term for different kinds of virtual machines such as **classic virtual machines**, **emulators,** and **containers**. Outside of this dissertation, the term virtual machine is often used for classic virtual machines.	
Virtual Network	**Conservation Strategy** "**Encapsulation**". A **virtual network** is created on top of a physical network such as the Internet and can consist of **virtual machines** and **emulations**. It abstracts network protocols and serves to isolate the **software-based artwork** from the **environment**. In combination with other **conservation strategies**, a virtual network aims to internalise external **dependencies** of a **software-based artwork**, and thus to stabilise the artwork.	
Web-based art	**Internet art** that uses the World Wide Web as artistic medium. In other words, artworks that are based on the http protocol (websites) constitute web-based art. **Net art** is usually understood as web-based art created in the 1990ies to the early 2000s. See also "**net art**" and "**internet-based art**"	
Work-defining properties	The properties of an artwork that provide a complete description of the artwork.	Based on (Laurenson 2014: 76).